UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA BIOMÉDICA

DISEÑO Y DESARROLLO DE UN SISTEMA DE ALINEACIÓN PARA SU UTILIZACIÓN EN ARTROPLASTIAS DE RODILLA

AUTOR: NAIARA FERRER RUIZ
TUTOR: FRANCISCO JAVIER SAIZ RODRÍGUEZ

Curso académico: 2018-19

AGRADECIMIENTOS

A mis padres y Ferran, por todo el apoyo y ánimos recibidos a lo largo de la realización del proyecto y por todas las veces que me han tenido que escuchar hablando sobre mi trabajo fin de grado. A mi tutor, Javier Saiz, por su ayuda en la elaboración del trabajo y sus consejos. A Jaime Mora y Joe Spencer por darme la oportunidad de aprender de ellos en Sensor City y confiar en mi para la realización de este proyecto.

RESUMEN

Las artroplastias de rodilla son las cirugías de prótesis de rodilla. Uno de los objetivos a conseguir en esta cirugía es que los huesos tibia y fémur queden correctamente alineados, para que el movimiento después de la operación sea el movimiento natural de la pierna, o al menos, lo más similar posible a este. Para realizar esta alineación, existen sistemas de navegación que parecen no ser fáciles de utilizar para los cirujanos, por ello, en el presente proyecto se ha diseñado un sistema sencillo para la detección de la posición del fémur y la tibia durante una cirugía de recambio de rodilla.

El sistema está formado por un dispositivo central compuesto por un Arduino y una pantalla que se comunica, mediante conexión Bluetooth, con dos dispositivos periféricos que cuentan con un Arduino y un sensor acelerómetro, giroscopio y magnetómetro cada uno. Estos están posicionados en puntos clave del fémur y la tibia y mandan sus respectivas posiciones al dispositivo central, para mostrar la alineación relativa de los huesos en la pantalla. De esta manera, el cirujano puede saber qué posición es la correcta en la colocación de la prótesis.

El dispositivo se ha diseñado para ser utilizado por los cirujanos ortopédicos, durante la operación, y mejorar los resultados de la misma mediante una mejor alineación de los huesos.

Palabras clave: rodilla, artroplastia, acelerómetro, giroscopio, magnetómetro, alineación, Arduino

RESUM

Les artroplàsties de genoll son les cirurgies de pròtesis de genoll. Un dels objectius a aconseguir en aquest cirurgia és que les ossos fèmur i tíbia queden correctament alineats, per a que després de l'operació el moviment sigui el moviment natural de la cama, o el mes similar possible. Per a realitzar aquesta alineació, existeixen sistemes de navegació que no són fàcils d'utilitzar per als cirurgians, per això, en el present projecte s'ha dissenyat un sistema senzill per a la detecció de la posició del fèmur i la tíbia durant una cirurgia de recanvi de genoll.

El sistema està format per un dispositiu central format per un Arduino i una pantalla que es comunica, mitjançant connexió Bluetooth, amb dos dispositius perifèrics, que estan formats per un Arduino i un sensor acceleròmetre, giroscopi i magnetòmetre cadascun. Aquests estan posicionats en punts clau del fèmur i la tíbia i envien les seues respectives posicions al dispositiu central, per a mostrar la alineació relativa dels ossos en la pantalla. D'aquesta manera, el cirurgià pot saber quina posició es la correcta en la col·locació de la pròtesis.

El dispositiu s'ha dissenyat per ser utilitzar per cirurgians ortopèdics, durant l'operació, i millorar els resultats de la mateixa mitjançant una millor alineació dels ossos.

Paraules clau: genoll, artroplàstia, acceleròmetre, giroscopi, magnetòmetre, alineació, Arduino

ABSTRACT

Knee arthroplasties are the knee prosthesis surgeries. One of the objectives to achieve in this surgery is the correct alignment of femur and tibia bones in order to obtain a natural movement of the leg, or at least, the most similar possible after surgery. Nowadays, to get that alignment, there are navigation systems that are not easy to use for surgeons, so in the present project a simple system to get the position of femur and the tibia during the replacement surgery has been designed.

The system is formed by a central device with an Arduino and a screen which communicate, by Bluetooth connection, with two peripheral devices with an Arduino, and an accelerometer, gyroscope and magnetometer sensor each one. These are positioned in key points of the bones, and send their respective positions to the central device, in order to show the relative alignment of the bones in the screen. Thus, the surgeon could know which is the correct position of the prothesis.

The device has been designed in order to be used by orthopedic surgeons, during the surgery, and to improve the results of it by better alignment of the bones.

Key Words: knee, arthroplasty, accelerometer, gyroscope, magnetometer, alignment, Arduino

ÍNDICE

DOCUMENTOS CONTENIDOS EN EL TFG

- Memoria
- Presupuesto
- Anexos

ÍNDICE DE LA MEMORIA

INDICE DEL PRESUPUESTO

ÍNDICE DE LOS ANEXOS

MEMORIA

CAPÍTULO 1. OBJETIVOS E INTRODUCCIÓN

1.1. OBJETIVOS

El objetivo de este trabajo es diseñar un dispositivo basado en un sistema de *tracking* que sea capaz de medir el ángulo entre dos puntos con una precisión de +/- 3 grados durante una artroplastia de rodilla. Además, este sistema debe ser sin cables, portátil y dotado de pantalla, de manera que permita un uso fácil y efectivo en el quirófano.

La idea del proyecto surge del doctor Sanjeev Agarwal, cirujano, consultor ortopédico, y director de la empresa Knee Innovations ltd (Reino Unido), que pone de manifiesto la necesidad existente entre los cirujanos de disponer de un dispositivo que utilizar durante las artroplastias de rodilla para calcular la alineación de los huesos largos que forman parte de la articulación de la rodilla, es decir, del fémur y la tibia. En estas cirugías, que consisten en sustituir tejido de la articulación por prótesis, es de gran importancia la correcta alineación de dichos huesos para la obtención de un buen resultado en cuanto a la futura movilidad de la pierna del paciente, por tanto, el objetivo de este dispositivo es aumentar el éxito de la operación y reducir la frecuencia de revisión quirúrgica.

Puesto que el dispositivo se va a utilizar en cualquier momento durante la operación, tiene que causar la mínima interferencia o incomodidad al cirujano y al equipo médico, por lo que la propuesta es un dispositivo sin cables, es decir, con batería incorporada, y con pantalla para facilitar la lectura de los datos.

Aunque este dispositivo ha sido diseñado para ser utilizado en artroplastias de rodilla, como realmente es un sistema de *tracking,* que determina la posición de cualquier objeto y el fundamento en el que se basa es la obtención del ángulo relativo entre dos puntos distantes, podrá ser utilizado en otras cirugías en las que sea necesaria una alineación de los miembros.

Las características finales de este dispositivo deben ser las siguientes:

- En primer lugar, debe ser capaz de medir el ángulo relativo entre dos puntos cercanos, en este caso, situados en el fémur y la tibia.
- En segundo lugar, el dispositivo no tiene que tener cables, para que su utilización no cause ningún inconveniente o interferencia durante una operación.
- En tercer lugar, debe incorporar una pantalla que suministre los datos.
- Finalmente, se desea que el dispositivo sea portátil y autónomo, es decir, que no tenga que estar conectado a la red eléctrica durante la cirugía.

1.2. INTRODUCCIÓN

En los últimos años ha aumentado sensiblemente la cantidad de intervenciones quirúrgicas del tipo de las artroplastias de rodilla, tanto primarias como de revisión. Este fenómeno, que no es exclusivo de España, puesto que se observa un patrón similar en otros países de Europa, está motivado por diversas razones: Es una cirugía bastante efectiva, con un notable porcentaje de éxito; los criterios para llevarla a cabo se han visto ampliados en los últimos tiempos y, además, se ha producido y se está produciendo un incremento constante en la incidencia de lesiones de la articulación de la rodilla, debido al progresivo envejecimiento de la población.

Si la cirugía tiene éxito el paciente podrá mover la articulación con total normalidad y el comportamiento de su miembro inferior será como el de cualquier persona sana, por tanto, no será necesario realizar una nueva intervención. Sin embargo, en caso contrario deberá realizarse una revisión quirúrgica pasado un tiempo. Esto ocurre con cierta frecuencia, los registros clínicos existentes muestran que entre los años 1997 y 2011 se llevaron a cabo 431.349 cirugías primarias de este tipo y 42.111 revisiones en España.

Además, estas revisiones quirúrgicas de la prótesis suponen un gran impacto para los costes sanitarios y para la salud de los pacientes porque presentan una alta tasa de complicaciones y las estancias hospitalarias son más prolongadas que cuando se trata de cirugía primaria (Castells, X., Comas, M., Guerrero, R., Espallargues, M., Allepuz, A., & Sabatés, S., 2014)

Por tanto, queda a la vista la importancia del éxito de este tipo de cirugía. Este depende de diversos factores, pero entre ellos reviste gran importancia el que los huesos componentes de la articulación de la rodilla (rótula, fémur y tibia) queden alineados correctamente después de la operación.

Para analizar la posición de los huesos y llevar a cabo la correspondiente alineación, actualmente existen sistemas de *tracking* o navegación óptica cuyo método de funcionamiento es calcular la posición 3D de unos sensores a partir de luz emitida o reflejada. Sin embargo, la mayoría de los cirujanos encuentran estos sistemas demasiado complicados y difíciles de utilizar, ya que incorporan una tecnología muy compleja que requiere una formación y entrenamiento específicos, que en muchas ocasiones no han recibido. Un ejemplo de esto es el caso de un importante hospital de Valencia, donde cuentan con el sistema de navegación OrthoPilot, de la marca Braun, pero el grado de utilización de este es muy reducido, ya que la mayor parte de cirujanos prefieren utilizar los sistemas intramedulares, en los que la obtención de medidas es más fácil, pero a su vez, estas son menos precisas y fiables.

Por ese motivo, se pedía un sistema de fácil utilización que únicamente requiriese la utilización de dos sensores a colocar en la tibia y el fémur, y una pantalla para poder leer los datos.

1.3. EMPRESA

El presente trabajo es un proyecto realizado en el grupo LCR 4.0 de Sensor City en el marco de un programa de prácticas en empresa.

Figura 1. Logo de la empresa. (Sensor City, 2019).

La empresa Sensor City, en Liverpool, tiene su campo de acción entre las pequeñas y medianas empresas, ofreciendo ayuda técnica (materiales y equipamiento) y conocimiento, con el objetivo de desarrollar y transformar las ideas previamente concebidas por los solicitantes, y así diseñar y construir un prototipo que cumpla con las especificaciones demandadas y dé respuesta a las expectativas generadas (Sensor City, 2019).

Este proyecto surgió de parte de la empresa Knee Innovations ltd. El director de esta empresa es el cirujano ortopédico y consultor médico Sanjeev Agarwal, quien buscaba disponer de un dispositivo capaz de medir la alineación relativa entre el fémur y la tibia durante una artroplastia de rodilla, para así poder disminuir el margen de error y con ello las revisiones quirúrgicas.

CAPÍTULO 2. ESTADO DEL ARTE

2.1. ANATOMÍA DE LA RODILLA

La articulación de la rodilla es la articulación sinovial más grande del cuerpo y una de las más complejas. Como cualquier articulación sinovial es la responsable de otorgar movilidad a algún miembro del cuerpo, en este caso al miembro inferior, y posee mecanismos para inmovilizarse cuando la acción que se requiera sea la de soporte. Esta articulación está formada por el fémur, la tibia y la rótula (García-Porrero, J. A., & Hurlé, J. M., 2013).

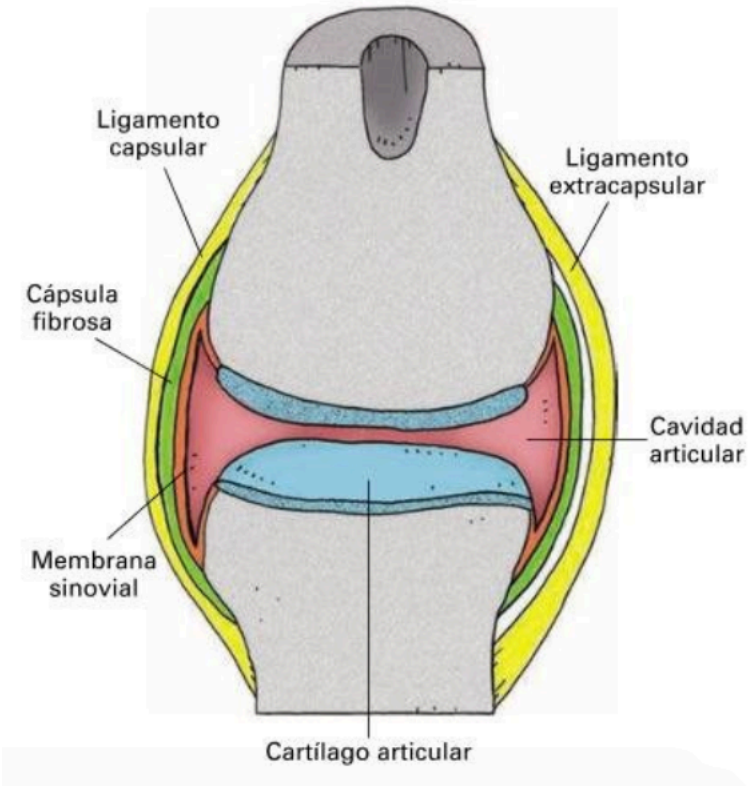


Figura 2. Articulación de la rodilla. (García-Porrero, J. A., & Hurlé, J. M., 2013)

El extremo distal del fémur se divide en dos masas óseas laterales, el cóndilo medial y el cóndilo lateral, que están separados posteriormente por la fosa intercondilea y unidos anteriormente en la superficie rotuliana, donde articulan con la rótula. Estos se encuentran recubiertos por cartílago hialino (García-Porrero, J. A., & Hurlé, J. M., 2013).

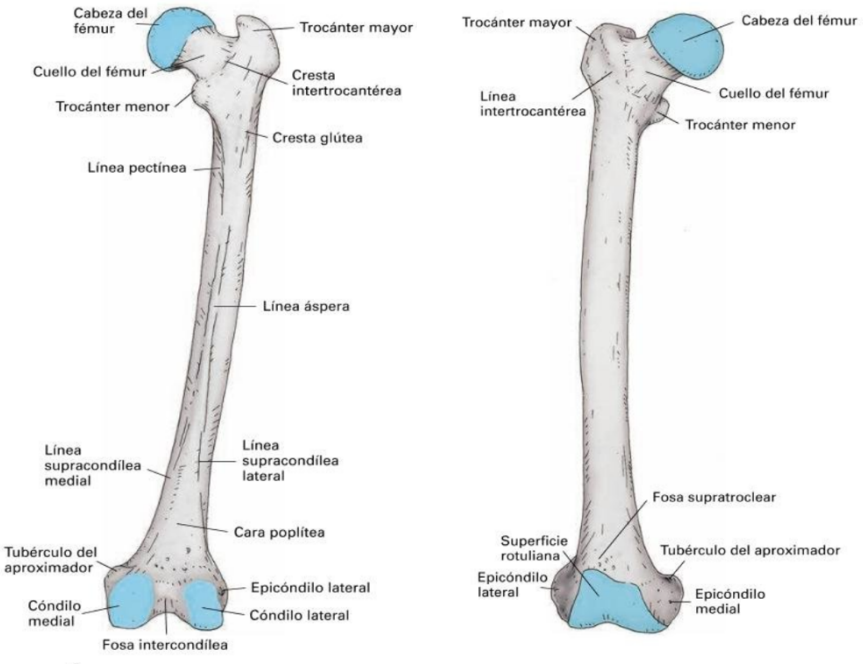


Figura 3. Fémur. (García-Porrero, J. A., & Hurlé, J. M, 2013).)

En cuanto a la tibia, en el extremo superior posee un cóndilo medial y un cóndilo lateral cuyas superficies están separadas por una región intercondílea, donde se insieren los ligamentos cruzados y los meniscos fibrocartilaginosos. Los dos meniscos fibrocartilaginosos, uno en cada lado, acomodan los cambios de forma de la superficie articular durante los movimientos de esta y son los encargados de distribuir las fuerzas de carga (García-Porrero, J. A., & Hurlé, J. M., 2013).

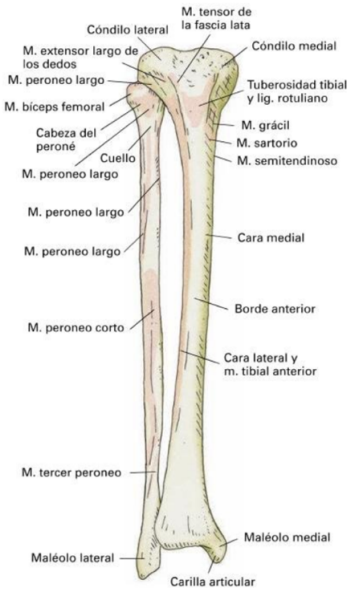


Figura 4. Tibia y peroné. (García-Porrero, J. A., & Hurlé, J. M., 2013)

Por otra parte, la rótula, también llamada patella, es el hueso sesamoideo más grande del cuerpo. Este se encuentra en el plano anterior de la rodilla, y es la encargada de que la contracción del cuádriceps sea más eficaz. Tienen forma triangular aplanada, y en su cara posterior se relaciona tanto con el fémur como con la tibia. Esta también está recubierta de cartílago hialino (García-Porrero, J. A., & Hurlé, J. M., 2013).

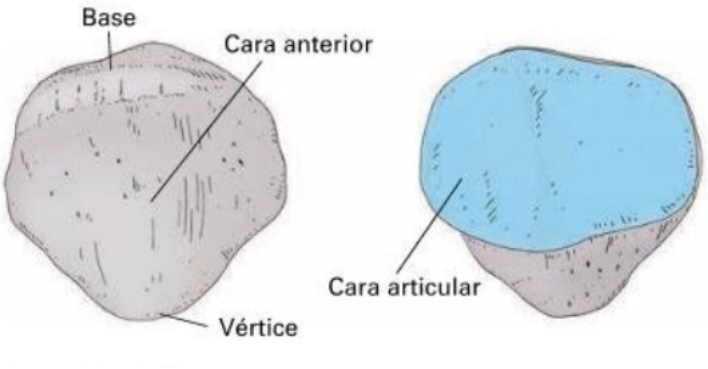


Figura 5. Rótula. (García-Porrero, J. A., & Hurlé, J. M., 2013)

Adicionalmente, esta articulación cuenta con los ligamentos cruzados, que están compuestos por fibras de colágeno tipo I, y debido a su elevada fuerza, dan estabilidad a la rodilla interconectando los finales adyacentes del fémur y la tibia para mantener las posiciones opuestas durante el movimiento.

Los movimientos de la rodilla se realizan de manera correlacionada en los tres planos anatómicos, es decir, en los planos sagital, transversal y coronal. En el plano sagital se puede considerar una articulación de tipo bisagra, la cual permite los movimientos de flexión y extensión, en un rango de 0 a 140 grados. En el plano transversal hay un movimiento de rotación, que se suma al movimiento anterior del plano sagital, para bloquear la articulación durante la marcha. Y, por último, en el plano coronal se produce la traslación, dirigido por los ligamentos cruzados (Gómez Vallejo, J., 2011).

Dado que la rodilla está relacionada con el soporte del peso, que atraviesa la extremidad inferior siguiendo el eje mecánico (línea que empieza en el centro de la cadera y concluye en el centro del tobillo), es de gran importancia que esté correctamente alineada para no alterar las fuerzas de carga y tensión de esta, para ello se utilizan los meniscos que, al aumentar la superficie de contacto, disminuyen la tensión (Drake, R., Vogl, W. and Mitchell, A).

2.2. PRINCIPALES TRASTORNOS O LESIONES DE RODILLA

La artropatía más frecuente en esta articulación es la enfermedad degenerativa articular. Esta puede ser resultado de una fuerza anormal a través de la articulación con un cartílago normal, o de una fuerza normal con un cartílago anormal.

Generalmente la enfermedad articular degenerativa ocurre en articulaciones sinoviales, como es la de la rodilla, y el proceso es llamado osteoartritis o artrosis. En las articulaciones afectadas por una osteoartritis, esta va perdiendo sus principales características anatómicas y motrices, lo que provoca dolor y movimiento limitado en dicha localización. Esto ocurre porque se produce una destrucción del cartílago que la protege.

Los hallazgos típicos cuando se da esta enfermedad incluyen una reducción del espacio articular, esclerosis articular, osteoporosis y formación de quistes en los huesos (Drake, R., Vogl, W. and Mitchell, A).

Figura 6. A. Articulación sana. B. Articulación con osteoartritis. (American academy of orthopaedic surgeons, 2019)

La etiología de la enfermedad degenerativa no es clara, pero hay algunos factores favorecedores, como la predisposición genética, el envejecimiento, el uso excesivo o la falta de uso de las articulaciones, y anormalidades metabólicas o nutricionales. Otros factores incluyen algún trauma articular o alguna enfermedad preexistente.

En un primer momento el tratamiento para la osteoartritis incluye cambios en el estilo de vida para prevenir el dolor, pero a medida que los síntomas progresan, un reemplazo de la articulación (artroplastia de rodilla) suele ser necesario, y aunque el reemplazo articular parece ser el gran remedio para este tipo de enfermedades, puede conllevar riesgos y complicaciones tales como infección y fallo a corto o largo plazo (Drake, R., Vogl, W. and Mitchell, A).

2.3. ARTROPLASTIA DE RODILLA

La artroplastia de rodilla es uno de los procedimientos con más éxito en el campo de la cirugía protésica del Sistema nacional de salud español. Esta implica una osteotomía del fémur y la tibia, dejando que la articulación central sea reemplazada por una prótesis. Es decir, consiste en reemplazar las superficies enfermas de la articulación de la rodilla para disminuir el dolor y restablecer la función y movimiento de dicha articulación y los tejidos que la controlan.

Esta cirugía está generalmente indicada en pacientes con artrosis que, en España, en cuanto a la rodilla, afecta alrededor del 14% de las mujeres y el 5,7% de hombres. (Castells, X., Comas, M., Guerrero, R., Espallargues, M., Allepuz, A., & Sabatés, S., 2014).

Solo en el año 2012, en este país se llevaron a práctica 42.451 intervenciones de este tipo, de las cuales el 90% fueron intervenciones primarias y el 10% revisiones, que se tienen que llevar a cabo cuando la prótesis no ha sido bien implantada o cuando ha pasado un tiempo, de entre 10 y 15 años, que suele ser la vida media de estas prótesis (Angulo Pueyo E, Ridao Lopez M, Martínez Lizaga N, Seral Rodríguez M, Bernal-Delgado E, 2014).

Asimismo, la variación de realización de artroplastias de rodilla entre los años y las comunidades autónomas es muy diversa. El instituto aragonés de Ciencias de la salud estudió que la probabilidad de ser operado de prótesis de rodilla tiene una gran dependencia con el área sanitaria de residencia del paciente, variando hasta 4 veces la probabilidad de artroplastia primaria y hasta 10 veces la probabilidad de revisión en ciertas áreas.

Además, estos también concluyeron que el número de operaciones de recambio de rodilla, en el periodo de 2002 a 2012 aumentó en un 47%, situación que está relacionada con el cambio en la edad de los pacientes intervenidos en 2012, ya que estos estaban en un rango de entre 51 y 86, mientras que en 2002 los pacientes estaban en un rango de edad de 53 a 84 años.

Conjuntamente, en este periodo de tiempo las revisiones se duplicaron, suponiendo entre el 8 y el 10% de las cirugías de reemplazo de rodilla (Angulo Pueyo E, Ridao Lopez M, Martínez Lizaga N, Seral Rodríguez M, Bernal-Delgado E, 2014).

Teniendo en cuenta los datos anteriores, se puede observar que esta es una cirugía muy recurrente, por lo que es de gran importancia su estudio y perfeccionamiento, por ello, a continuación, se va a analizar la técnica en profundidad.

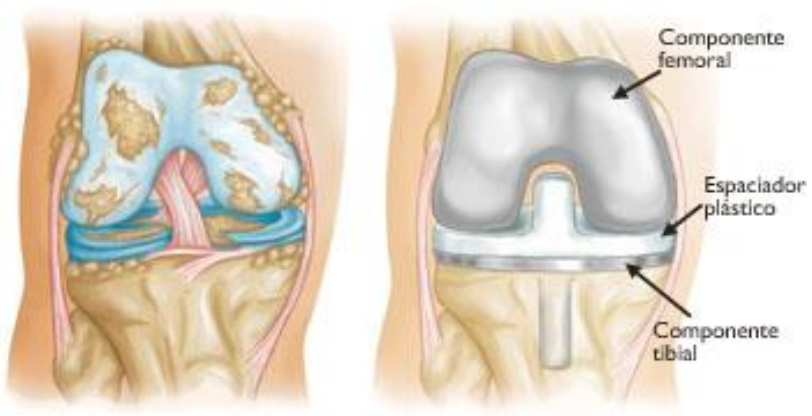


Figura 7. A. Articulación con osteoartritis. B. Articulación reemplazada por prótesis. (American academy of orthopaedic surgeons, 2019)

Esta intervención se ha realizado de diferentes maneras a lo largo de la historia. La primera artroplastia de rodilla, realizada por Verneuil en 1863, consistió en implantar una prolongación de la cápsula articular entre las superficies del fémur y la tibia para impedir que se fijaran después de haber sido resecadas. Las artroplastias modernas empezaron a realizarse en 1951 cuando Walldius desarrolló una prótesis en bisagra, hecha de resina acrílica, con una parte femoral y una tibial unida por una varilla de acero. Más tarde, al comprobar que la resina no era suficientemente resistente se empezó a utilizar cromo-cobalto.

Fue a partir de 1971 cuando comenzó a utilizarse una prótesis de baja fricción que consistía en dos superficies de acero, articuladas con una superficie de polietileno de alta densidad, y cementadas al hueso con polimetilmetacrilato. Esta permitía flexiones de más de 100º con una baja resección ósea.

En cuanto a los materiales, actualmente los más utilizados para las superficies articulares del fémur son el Vitallium (compuesto un 30% de cromo y 7% cobalto, molibdeno y níquel entre otros materiales), las superaleaciones de cobalto y la aleación Ti6Al4V. Para la parte tibial el material que se suele utilizar es cromo-cobalto (Rodríguez, J. L. M., Ochoa, D. R. H., Henríquez, J. A. V., & Méndez, 2012).

Respecto a la superficie sustituida, hoy en día existen tres tipos diferentes de prótesis: Unicompartimentales, bicompartimentales y tricompartimentales.

- Las unicompartimentales sustituyen la superficie de apoyo del fémur, tibia o rótula en cualquier compartimento de la articulación (lateral, medial o patelofemoral), y los otros dos huesos se dejan como estaban anteriormente.
- Las prótesis bicompartimentales reemplazan las superficies solo del fémur y la tibia tanto en el compartimento medial como en el lateral.
- Y finalmente, las tricompartimentales, reemplazan las superficies articulares de los tres huesos componentes de la articulación.

Las más utilizadas en España son las prótesis tricompartimentales, ya que estas resultan más cómodas y eficaces para el paciente, puesto que el movimiento se ve facilitado porque todas las superficies en contacto están diseñadas para encajar perfectamente unas con otras, y son construidas con materiales específicos que se ha comprobado que deslizan de forma óptima, por tanto, acarrearán menos problemas en el futuro.

Pese a que, como se ha dicho anteriormente, en el campo de la cirugía prostética los recambios de rodilla son los que más éxito tienen, este depende de muchos factores, incluyendo la selección del paciente, el diseño de la prótesis, el balance de tejidos blandos y la técnica quirúrgica.

Con relación a la técnica quirúrgica un factor de elevada importancia es la alineación de la pierna y la restauración de la línea de la articulación. Es decir, la alineación axial correcta es muy importante para la longevidad del implante, ya que una pequeña desviación de la posición puede llevar a una disminución de la funcionalidad y a un pronto aflojamiento de la prótesis. Y, por otra parte, la mala alineación de los componentes en cualquiera de los tres planos anatómicos puede llevar a mayores complicaciones como dislocaciones de rótula (Lavernia, C., & Alcerro, J., 2008).

Para que la alineación de los componentes protésicos sea la correcta es de vital importancia realizar los cortes óseos de una manera determinada, para lo que existen diferentes métodos. El más utilizado es el método clásico, que utiliza guías intramedulares para realizar los cortes femorales y guías extramedulares para realizar los cortes tibiales. Esta es bastante subjetiva.

Por eso, hoy en día, los hospitales cuentan con novedosos sistemas de navegación que aportan información como la alineación de la prótesis o la situación de los tejidos blandos, pero los cirujanos no los suelen utilizar de manera ordinaria por los motivos que ya se han explicado anteriormente.

2.3.1. Procedimiento quirúrgico con alineación intramedular y extramedular

En cualquier tipo de artroplastia de rodilla, antes de empezar la intervención, el cirujano examina las imágenes radiológicas que se habrán tomado al paciente previamente, para determinar la situación de la articulación antes de la operación.

Cuando ya está todo listo y el paciente anestesiado, se realiza una isquemia, normalmente con manguito, en la extremidad a operar para evitar que durante la cirugía circule sangre por ella y así favorecer la visión de todos los componentes articulares.

El primer paso de la cirugía es abrir la articulación de la rodilla y realizar el orificio, para introducir la guía intramedular. Este tiene que realizarse sobre unos 12 mm anterior al techo de la fosa intercondílea y 1,5 mm medial al centro (Hinarejos Gómez, Pedro.).

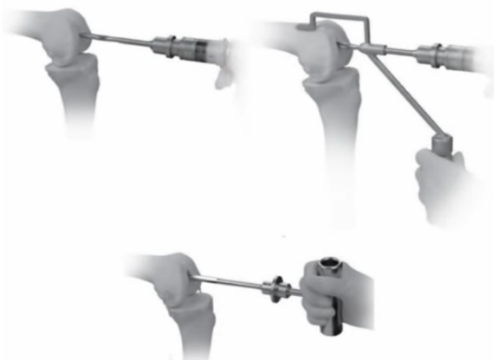


Figura 8. Agujereado e introducción de la guia femoral. (PROTESIS TOTAL DE RODILLA -Instrumental y técnica- Dr.Gonzalo Mora, 2012)

Con la guía ya dentro del hueso, se adhiere a ella, perpendicularmente a la guía, una plantilla que lleva incorporados diferentes agujeros por donde pasar el instrumento para realizar la resección del hueso.

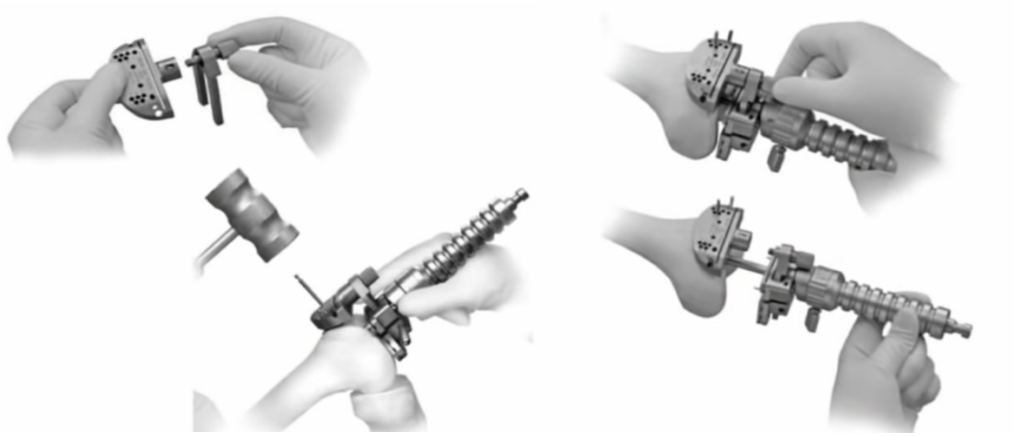


Figura 9. Incorporación de las plantillas. (PROTESIS TOTAL DE RODILLA -Instrumental y técnica- Dr.Gonzalo Mora, 2012)

Por tanto, siguiendo la plantilla, se cortan las partes correspondientes del fémur. Como se está utilizando una guía intramedular que sigue el eje anatómico del fémur, se tiene que realizar el corte con un ángulo valgo, que varia dependiendo del paciente (Hinarejos Gómez, Pedro).

A continuación, se realiza un procedimiento similar en la tibia, pero en este caso la guía es una barra de alineación extramedular que sigue el eje mecánico de la tibia. Se inmoviliza la barra, cuya extensión empieza en el extremo proximal de la tibia termina en la parte distal del hueso y se adjuntan las plantillas para realizar un corte perpendicular al eje mecánico.

Figura 10. Colocación de la guía tibial. (PROTESIS TOTAL DE RODILLA -Instrumental y técnica- Dr.Gonzalo Mora, 2012)

Figura 11. Colocación de la guía tibial. (PROTESIS TOTAL DE RODILLA -Instrumental y técnica- Dr.Gonzalo Mora, 2012)

Con todos los cortes correctamente realizados, el cirujano procede a introducir unas prótesis provisionales, que deben encajar en los huesos resecados, y comprobar que el movimiento de la articulación es el correcto. Una vez verificado, introduce y cementa las prótesis definitivas en los huesos resecados.

Para terminar, se elimina el manguito de la extremidad para que vuelva a circular la sangre con normalidad y se sutura.

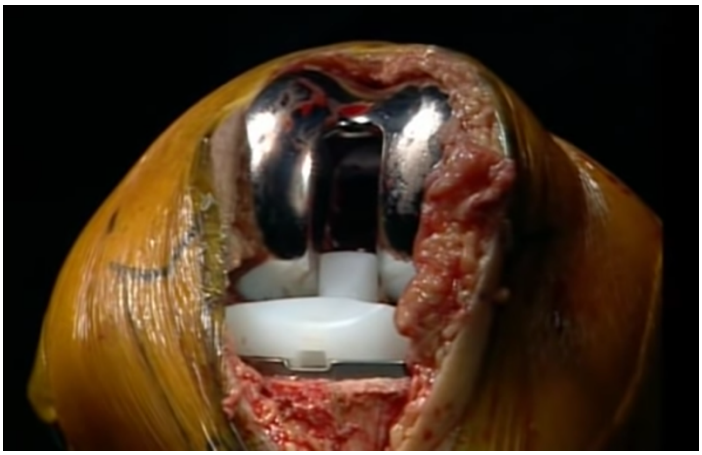


Figura 12. Finalización de la cirugía. (PROTESIS TOTAL DE RODILLA -Instrumental y técnica- Dr.Gonzalo Mora, 2012)

Como se ha visto, esta técnica utiliza la alineación mecánica con la que se obtiene una línea recta entre los centros de la cabeza del fémur, la rodilla y la cúpula astragalina (Hinarejos Gómez, Pedro). Este tipo de cirugía lleva realizándose durante muchos años obteniendo buenos resultados. Sin embargo, el principal problema de esta técnica es que no se dispone de un sistema intraoperatorio que indique la orientación exacta de los cortes de resección o de colocación de las barras lo que en muchas ocasiones produce una mala alineación que conlleva a cirugías de revisión.

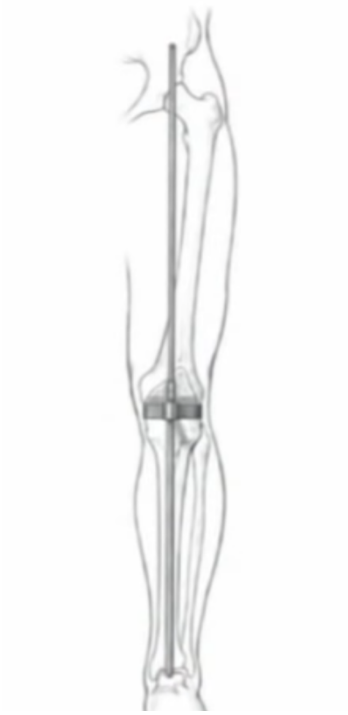


Figura 13. Alineación de las prótesis. (PROTESIS TOTAL DE RODILLA -Instrumental y técnica- Dr.Gonzalo Mora, 2012)

2.3.2. Procedimiento quirúrgico con sistema de navegación

Para llevar a cabo esta técnica quirúrgica la operación sigue el mismo curso que en la técnica clásica hasta el momento de empezar la resección. Pero en este caso, a partir de ese momento se va a utilizar un sistema de navegación o tracking. Este es un sistema utilizado para medir el cambio en tiempo real en la posición y orientación de un objeto de tres dimensiones.

Uno de los sistemas de navegación utilizados en España para artroplastias de rodilla es Orthopilot, de la marca Braun, que se desarrolló para conseguir una perfecta alineación del implante quirúrgico sin necesidad de realizar ninguna prueba preoperatoria ni utilizar rayos (TAC y RM) lo que, a parte de ser beneficioso para el paciente, también reduce el coste de la cirugía. Aunque cabe decir, que muchos cirujanos que lo utilizan siguen realizando estas pruebas preoperatorias por seguridad.

Este sistema se puede utilizar para artroplastias totales o parciales, para procedimientos menos invasivos e incluso para revisiones posteriores a la primera cirugía. Permite realizar un estudio cinemático intraoperatorio para calcular el eje mecánico de la rodilla (o de la cadera, si se utiliza para ese tipo de cirugía) y la situación de los cortes de resección. Para llevar a cabo este procedimiento la cámara del sistema rastrea el movimiento de los sensores fijados al hueso (Orthopilot. Bbraun sharing expertise).

.

Figura 14. (Orthopilot. Bbraun sharing expertise.)

En este tipo de cirugía, se realiza una alineación cinemática, en la que se respeta el eje de flexión de la rodilla, que es diferente en cada paciente, para lo que se realiza el estudio cinemático, del que se ha hablado anteriormente, seguido de una palpación de los puntos de referencia anatómicos del hueso. En la pantalla se pueden observar los pasos a seguir, lo que facilita la ejecución al cirujano.

Una vez hecho esto, se realiza la resección tibial, de manera perpendicular al eje cinemático del paciente, con una plantilla que se atornilla al hueso del paciente. En este momento, los mismos sensores utilizados anteriormente se pueden colocar en esta plantilla para poder conocer su posición concreta (Hart, R., Janeček, M., Chaker, A., & Buček, P., 2003).

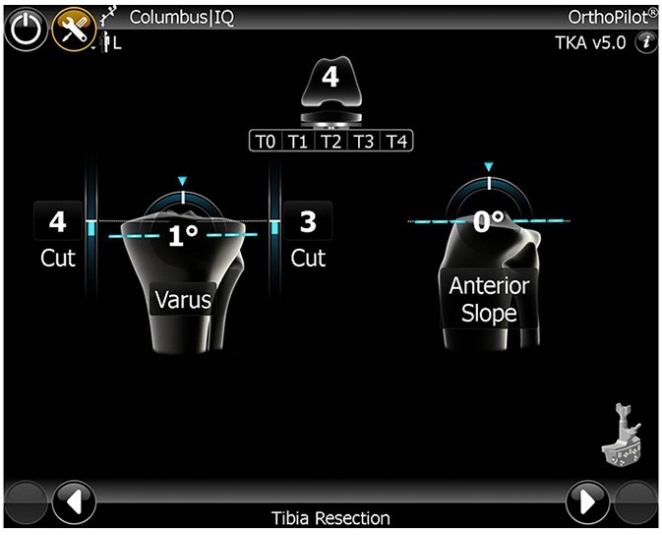


Figura 15. (Orthopilot. Bbraun sharing expertise.)

Entonces se calcula la distancia articular, es decir, la cantidad de tejido blando que hay en la articulación, midiendo la distancia medial y lateral de ambos huesos tanto en flexión como en extensión, y se muestra en pantalla.

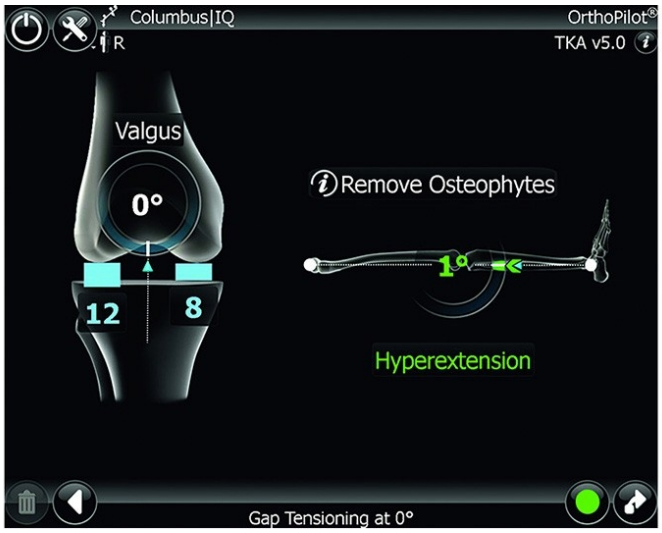


Figura 16. (Orthopilot. Bbraun sharing expertise.)

Con todo lo anterior se hace una planificación de la resección del fémur teniendo en cuenta el tamaño y la posición del hueso. Cuando la planificación está hecha, la pantalla nos muestra el resultado ideal de alineación en la situación dada.

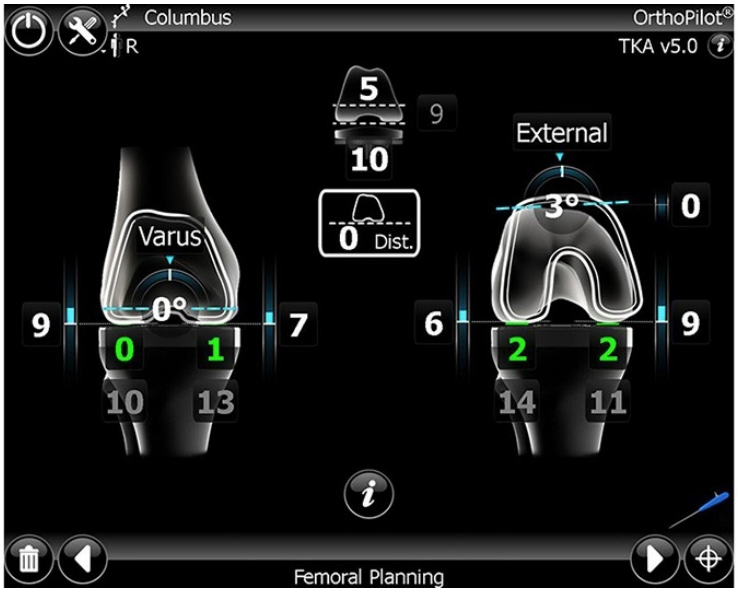


Figura 17. (Orthopilot. Bbraun sharing expertise.)

Entonces, se realiza la resección del fémur de la misma manera que anteriormente se había hecho la de la tibia, e introduciendo los sensores en los puntos iniciales se valida, ya que en la pantalla se puede ver el resultado final.

Una vez se han realizado las resecciones y se ha decidido el tamaño de las prótesis que se van a poner, se colocan unas prótesis provisionales para comprobar que, efectivamente, el movimiento se realiza correctamente y esa medida y localización son las que corresponden. Cuando ya se ha validado, se introducen las prótesis definitivas cementándolas al hueso y se siguen los mismos pasos que en el método clásico.

Varios estudios han demostrado que este es un sistema totalmente válido para su utilización en artroplastias tanto de rodilla, como de cadera, y que sus resultados a largo plazo son mejores debido a su alta precisión. No obstante, por la dificultad que encuentran, tanto en este como en otros sistemas de navegación, la mayoría de los cirujanos sigue utilizando la técnica clásica explicada anteriormente (Clemens, U., & Miehlke, R. K., 2003). Por ello, es de gran importancia mejorar la técnica de alineación para los procedimientos realizados con guías extramedulares e intramedulares, y para ello se ha llevado a cabo el presente proyecto.

CAPÍTULO 3. DISEÑO DEL SISTEMA

3.1. DESCRIPCIÓN GENERAL DE FUNCIONAMIENTO DEL SISTEMA

Antes de explicar a fondo el desarrollo del proyecto, se va a describir de forma general el sistema que se va a diseñar, para así tener una visión global del estudio y explicar sobre él las diferentes partes que lo forman.

Como se ha descrito en la parte de objetivos, el propósito del proyecto es diseñar un dispositivo capaz de medir la alineación perioperativa de la articulación en los planos coronal, sagital y transversal durante una artroplastia de rodilla, antes de cementar la prótesis al hueso. Esta información se mostrará en una pantalla.

Este dispositivo se utilizaría en las cirugías realizadas con guías intramedulares para el fémur y guías extramedulares para la tibia, lo que ayudará al cirujano a tener conocimiento sobre los ángulos exactos a los que está haciendo las resecciones y colocando las prótesis. No obstante, este es un anteproyecto en el que el dispositivo no se ha diseñado con suficiente miniaturización (12mm) como para poderse introducir dentro del fémur, y por tanto está planificado para utilizarse exteriormente tanto en tibia como en fémur.

A continuación, se muestra un diagrama de esta funcionalidad:

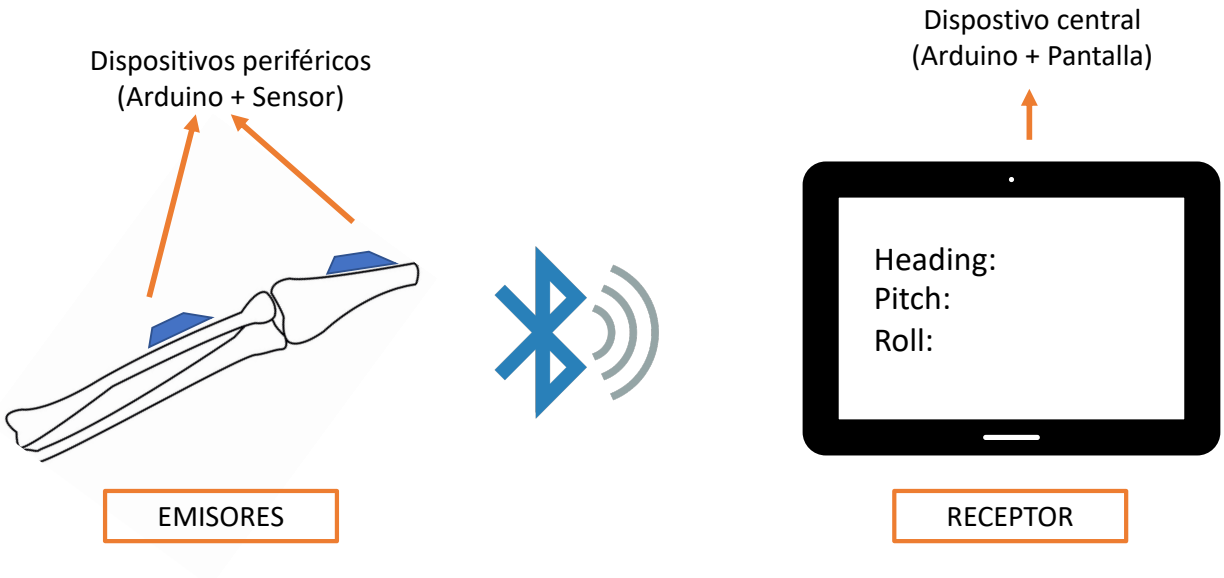


Figura 18. Diagrama del dispositivo.

Por tanto, el proyecto se lleva a cabo en tres fases diferenciadas.

- La primera fase es el desarrollo del dispositivo electrónico, que a su vez está dividido en otras dos partes que se describen a continuación.

  Por una parte, hay dos dispositivos periféricos dispuestos en el fémur y la tibia, formados cada uno de ellos por un sensor acelerómetro, giroscopio y magnetómetro y un Arduino. Los sensores detectan la posición del punto donde están colocados respecto al norte y la envían mediante conexión I2C (*Inter – Intergrated Circuit)* al Arduino.

  Seguidamente, el Arduino envía la información recogida vía BLE (Bluetooth de bajo consumo), a un dispositivo central.

  Por otro lado, está el dispositivo central mencionado anteriormente, compuesto por una placa Arduino, que recibe los datos desde los dispositivos periféricos por conexión Bluetooth, y una pantalla. El software del Arduino, cuando recibe los datos de los dispositivos periféricos, obtiene la diferencia entre el ángulo de la tibia y el ángulo del fémur y lo manda, por conexión SPI (*Serial peripheral interface*) a la pantalla que muestra el ángulo relativo.

- La segunda fase es el firmware, que se trata del software que interactúa directamente con el hardware y controla los circuitos electrónicos del dispositivo descrito anteriormente. Por tanto, este es el código que se implementa en las diferentes placas, y se explicará detalladamente en el apartado 3.3.

- Por último, se realiza un diseño y posterior impresión 3D de envolturas físicas para los dispositivos electrónicos.

3.2. HARDWARE

3.2.1. Descripción general del hardware

Esta sección está dedicada a la parte electrónica del presente proyecto, es decir, al desarrollo de un circuito en el que se integrarán los diferentes componentes del sistema.

Principalmente, la función de este dispositivo es adquirir señales físicas desde dos receptores situados en diferentes puntos, transmitirlas a un dispositivo central inalámbricamente, compararlas y mostrarlas en una pantalla.

Si se centra la atención en el flujo de información se concluye que los dispositivos periféricos han de tener una etapa que reciba la señal física (sensores) y una etapa que envíe los datos (módulo Bluetooth). Mientras que el dispositivo central deberá tener una etapa que reciba los datos, una etapa que los interprete y, por último, una etapa que los muestre (pantalla).

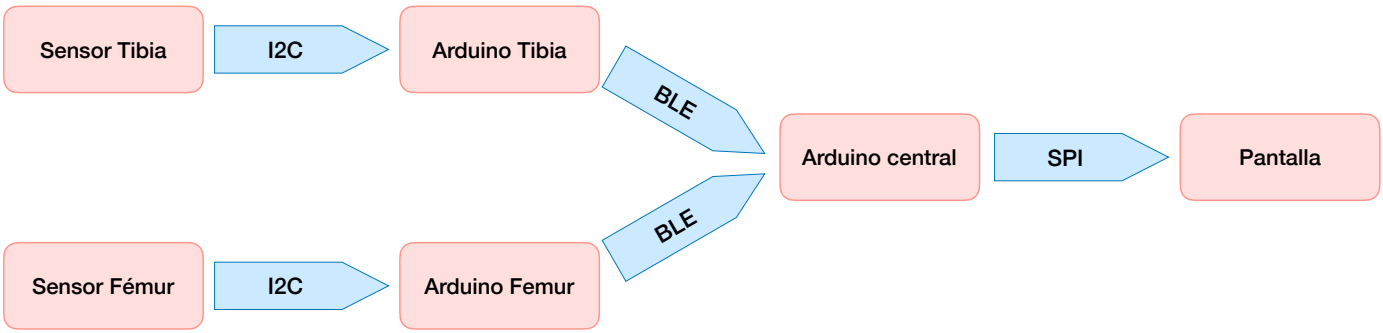


Figura 19. Descripción de funcionamiento del dispositivo.

3.2.2. Descripción de las funciones del sistema

A. Detección de la posición/orientación

Un objeto en movimiento en un espacio tridimensional, como es el mundo real, tiene 6 grados de libertad, tres para traslación y tres para rotación. Si se utiliza un sistema de coordenadas cartesiano, las traslaciones son los ejes X, Y y Z, y las rotaciones en esos ejes se llaman *yaw* o *heading, pitch* y *roll* respectivamente. Esto define un conjunto de datos de 6 números que necesitan ser medidos con suficiente velocidad, ya que el objeto puede estar moviéndose rápidamente. Los sistemas capaces de medir estos datos se llaman sistemas de tracking o de detección de posición.

En este proyecto se necesitará un sistema de *tracking,* puesto que se quiere medir la orientación de un dispositivo colocado en la tibia frente a un dispositivo colocado en el fémur.

Existen diferentes metodologías para la obtención de este conjunto de datos, como se va a explicar a continuación.

Métodos mecánicos

Los detectores de posición mecánicos consisten en estructuras cinemáticas en las que hay diversos puntos sensorizados interconectados entre ellos. Las distancias entre los puntos se conocen a priori y, por tanto, pueden ser utilizadas por un modelo cinemático informatizado que, mediante procedimientos de cálculo y representación gráfica, permite la determinación de la posición y orientación de dos estructuras cinemáticas distintas.

Estos son simples y fáciles de utilizar, y su precisión depende básicamente de la resolución de los sensores utilizados. Además, no presentan interferencias frente a materiales metálicos o

magnéticos ni tienen que encontrarse en un campo de visión determinado para poder ser utilizados.

Sin embargo, estos métodos son demasiado invasivos, ya que deben colocarse grandes aparatos en el paciente, por lo que no dejan suficiente libertad de movimiento para utilizarse en cirugía, si bien cabe decir que se han utilizado anteriormente sin demasiado éxito (Burdea, G. C., & Coiffet, P., 2003).

Por tanto, se van a estudiar otros métodos posibles para su utilización en este proyecto. Los siguientes no presentarán el problema de ser invasivos, ya que se tratan de métodos de medida sin contacto.

Métodos magnéticos

Los detectores magnéticos son dispositivos de medida sin contacto. Estos utilizan el campo magnético producido por un transmisor inmóvil para determinar la posición en tiempo real de un elemento, llamado receptor, en movimiento. El transmisor consiste en tres antenas formadas por tres bobinas ortogonales dispuestas en un cubo ferromagnético, que son excitadas secuencialmente para producir tres campos magnéticos ortogonales que penetran en el receptor produciendo una señal en forma de voltaje. Estos voltajes son muestreados por una unidad electrónica que utiliza un algoritmo de calibración para determinar la posición/orientación del receptor en relación con el transistor.

Sin embargo, la precisión de este tipo de dispositivo es muy baja cuando tiene cerca objetos metálicos o magnéticos, por tanto, objetos de esta índole deberían ser eliminados de la proximidad de los sensores, sino serían necesarias medidas complementarias y una calibración mucho más compleja (Burdea, G. C., & Coiffet, P., 2003).

Métodos ultrasónicos

Estos utilizan una señal ultrasónica producida por un transmisor estacionario para determinar la posición en tiempo real de un objeto receptor en movimiento.

De la misma manera que en los métodos magnéticos, se tienen tres componentes: un transmisor, un receptor y una unidad electrónica. La diferencia es que en este caso no se va a producir ninguna interferencia por la cercanía de materiales metálicos, ya que el transmisor es un set de tres altavoces ultrasónicos colocados a una distancia determinada entre ellos, y el receptor un set de tres micrófonos colocados también de manera triangular en una plataforma rígida y pequeña, que se inmoviliza en el objeto a monitorizar.

Como la velocidad del sonido en el aire cambia con la temperatura, si la temperatura de una habitación es conocida se puede calcular la velocidad y utilizarla para medir las distancias basadas en el tiempo de “vuelo” entre el receptor y el emisor mediante triangulación. Se

miden un total de nueve distancias para determinar la posición y orientación del plano que contiene los tres micrófonos, la unidad de control muestra los micrófonos, convierte las lecturas en posición y orientación basadas en constantes de calibración, y transmite los datos al ordenador, donde se puede visualizar la representación gráfica.

Sin embargo, la tasa de actualización es bastante baja, y si se tienen que monitorizar diferentes objetos (como en este caso, los huesos fémur y tibia), hay que utilizar más receptores, lo que significa incorporar multiplexado y reducir más la tasa de actualización.

Por otra parte, es necesario un campo visual directo entre el transmisor y el detector, situación que no se va a poder dar en el caso de utilizarla para cirugía (Burdea, G. C., & Coiffet, P., 2003).

Métodos ópticos

Los métodos ópticos utilizan sensado óptico para determinar la posición/orientación en tiempo real de un objeto. Estos también aplican el principio de triangulación, pero presentan la ventaja de que su tasa de registro y actualización es mucho más alta que en los métodos ultrasónicos.

Normalmente, consisten en un escáner láser fijo con dos haces rotatorios y un sensor móvil que puede ser unido al objeto a monitorizar. El escáner proyecta dos haces infrarrojos que alcanzan el elemento sensor, el rayo reflejado se registra en un detector de fotodiodos, que controla el tiempo y los ángulos.

Este es el método que utilizan los sistemas de navegación que existen actualmente en el mercado, ya que su precisión es alta y su tasa de actualización también. Sin embargo, su principal inconveniente es la necesidad de un campo directo de visión (Burdea, G. C., & Coiffet, P., 2003).

Métodos inerciales

Los dispositivos inerciales son sensores independientes que miden la tasa de cambio en la orientación de un objeto y en la velocidad de traslación del mismo.

Frente a los anteriores, estos ofrecen la ventaja de no necesitar un transmisor para funcionar y, por tanto, no necesitar campo de visión directo.

La tasa de cambio en la orientación de un objeto o la velocidad angular está medida por giroscopios basados en el principio de la fuerza de Coriolis. Tres giroscopios son posicionados en ejes ortogonales midiendo las velocidades angulares ($\omega = \frac{d\theta}{dt}$).

Esta velocidad angular se integra respecto al tiempo para obtener el ángulo de orientación en los tres ejes de la siguiente manera:

$$\theta(t) = \int_0^t \omega(t) d(t) \approx \sum_0^t \omega(t) * T_s$$

La tasa de cambio en la velocidad de traslación o aceleración se obtiene utilizando tres acelerómetros que son incorporados con los tres giroscopios para medir aceleraciones. Sabiendo la orientación del objeto por el giroscopio y sustrayendo la aceleración gravitacional se puede obtener la aceleración del objeto, siendo la posición finalmente obtenida por una doble integral respecto al tiempo, siempre teniendo en cuenta la posición de inicio del objeto, es decir, con calibración inicial.

Estos pueden presentar pequeños errores en el acelerómetro debidos a ruido de alta frecuencia, como por ejemplo las vibraciones, pero son sencillos de eliminar utilizando el llamado filtro complementario, que es muy simple y efectivo. El filtro complementario combina los datos de baja frecuencia del acelerómetro y los datos de alta frecuencia del giroscopio mediante la siguiente ecuación:

$$Ángulo = 0{,}98 * (Ángulo + (datos\ del\ giroscopio * dt) + 0{,}02 * (datos\ del\ acelerómetro)$$

Además, tienen un error de apilamiento que surge porque los datos del giroscopio cambian con mayor rapidez que la frecuencia de muestreo y, por tanto, crece proporcionalmente con el tiempo, acumulándose y provocando que el valor calculado no sea el real. Pese a ello, se conoce que la solución es utilizar, además de los datos del sistema de medida inercial, datos de otro tipo de *tracker,* para reiniciar periódicamente el output del sistema inercial. De esto surgen los sistemas híbridos (Burdea, G. C., & Coiffet, P., 2003).

Métodos híbridos

Como se ha dicho anteriormente, estos son los que utilizan dos o más sistemas de detección de posición de manera que la detección sea más precisa que con una sola tecnología.

Cuando solo se necesitan datos de orientación, la principal fusión se realiza con magnetómetros alineados con los tres giroscopios. En este caso, los datos de los tres magnetómetros pueden ser utilizados para determinar el norte magnético local, que no varía con la posición del objeto que se está detectando. Esto compensa el error en la medida del *heading*, es decir, en el plano horizontal. Sin embargo, cualquier campo magnético local como un metal afectará a la lectura de los magnetómetros (Burdea, G. C., & Coiffet, P., 2003).

Una posibilidad para reducir los probables errores es la utilización del filtro de Kalman. Este filtro se usa siempre que haya incertidumbre en la información dada por un sistema dinámico

en el que se puede suponer que sus dos variables son aleatorias y distribuidas de forma Gaussiana (Faragher, R., 2012).

Se trata de un conjunto de ecuaciones que proporciona una solución óptima por el método de los mínimos cuadrados. Se describe el sistema con un conjunto de variables de estado, que contienen toda la información en un punto concreto en el tiempo. Con esta información, se predice el futuro comportamiento del sistema utilizando el comportamiento del pasado (Ramírez, Á. S., 2003).

Estima el proceso utilizando control de retroalimentación. Calcula el estado del proceso en un punto y obtiene *feedback* de los datos observados. Así, se tienen dos ecuaciones, las de medida, que monitorizan el estado actual y predicen el siguiente, y las de actualización de tiempo, que se actualiza dependiendo de la matriz de predicción y por tanto pueden considerarse de pronóstico (Welch, G., & Bishop, G., 1995).

De esta manera, se tiene un algoritmo de pronóstico – corrección que pronostica un nuevo estado y su incertidumbre corrigiendo con una nueva medida (Ramírez, Á. S., 2003).

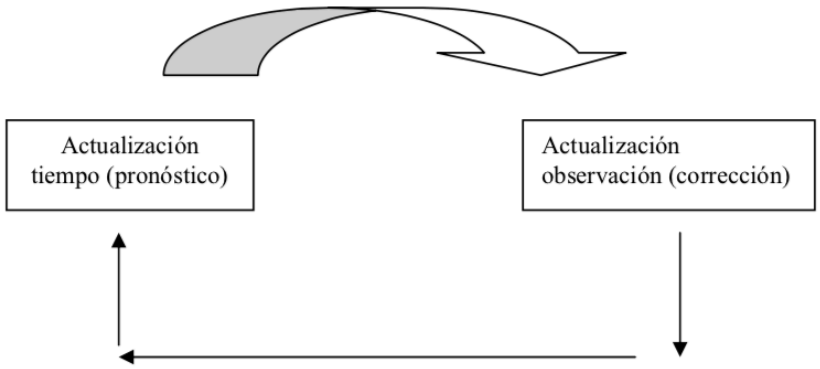


Figura 20. Funcionamiento del filtro de Kalmar (Ramírez, Á. S., 2003)

Por tanto, teniendo en cuenta las características de los métodos estudiados, se llega a la conclusión de que el método de preferencia a utilizar en este proyecto para la detección de la orientación relativa de la tibia frente al fémur es el método híbrido inercial con magnetómetros.

B. Microprocesador

Es necesario utilizar un microprocesador en la parte femoral y tibial del dispositivo, para recoger los datos del sensor, procesarlos y poder mandarlos para visualizarlos. De igual forma, se necesita otro microprocesador en la parte de la pantalla para poder recibir y tratar los

datos. Los dispositivos que pueden ser utilizados como microprocesador, en este caso, por sus características, serían la Raspberry Pi y el Arduino.

Raspberry Pi

Raspberry Pi es un pequeño ordenador con software Linux que se suele utilizar para aprender a programar y construir proyectos de hardware y software. Esta plataforma es de código abierto, y permite controlar componentes Electrónicos (Raspberry Pi, 2019).

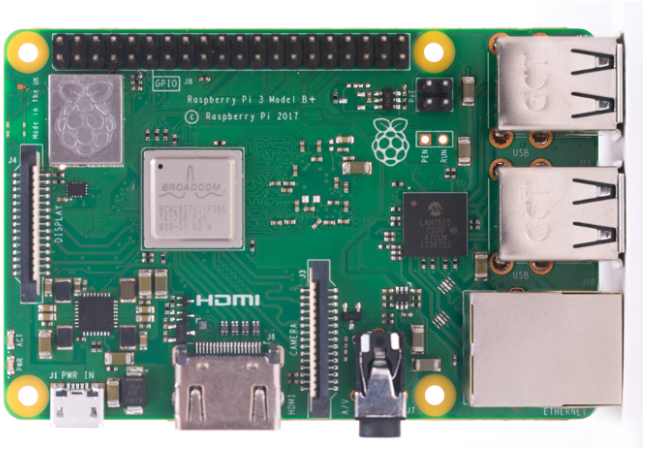


Figura 21. Placa de Raspberry Pi (Raspberry Pi, 2019).

Estas se pueden programar con diferentes softwares, siendo el oficial Raspbian, y con diferentes lenguajes, aunque el más utilizado para poder controlar las GPIO, es decir, los pines que controlan los componentes electrónicos, es Python.

Arduino

Por otra parte, Arduino es una plataforma de código abierto basada en hardware y software de fácil uso. Esta plataforma se creó como una herramienta fácil dirigida a estudiantes para hacer prototipos y se ha ido adaptando a nuevas necesidades y retos (Arduino, 2019).

Las placas de Arduino se pueden conectar a módulos adicionales como, por ejemplo, sensores, para aumentar sus características técnicas.

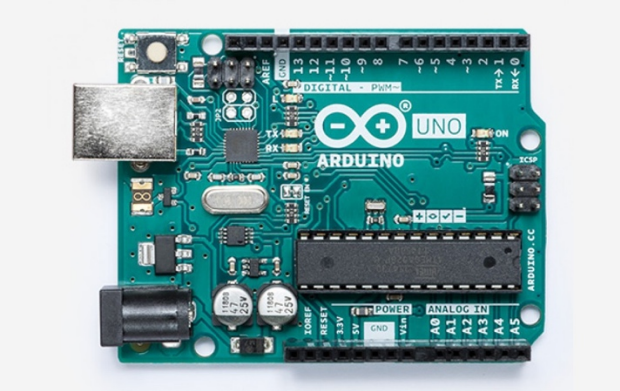


Figura 22. Placa de Arduino (Arduino, 2019).

Para ser utilizadas, estas placas se tienen que programar con el software Arduino IDE (Entorno de Desarrollo Integrado), que está compuesto por un editor de texto, un espacio de avisos (donde se notifica si hay algún error en el programa), y los diversos botones que permiten guardar, cargar, etc.

El lenguaje utilizado en este software es C++, que es un lenguaje imperativo orientado a objetos derivado del C.

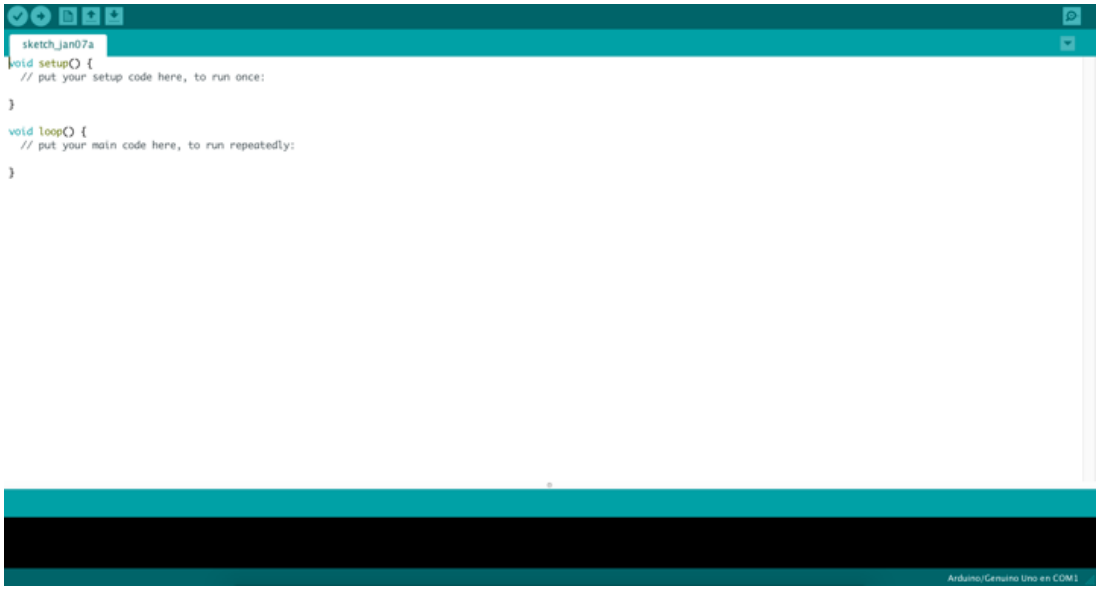


Figura 23. Arduino IDE (Arduino, 2019).

Además de las placas oficiales, actualmente también se pueden encontrar placas de otras marcas, como por ejemplo Adafruit, que tienen diversas funcionalidades.

Debido al amplio abanico de usos que se les puede dar a las placas de Adafruit, y al fácil lenguaje y utilización de Arduino se decidió utilizar este tipo de dispositivo para el presente proyecto.

C. Conexión inalámbrica

Para el funcionamiento del dispositivo diseñado resulta imprescindible almacenar los datos de los sensores dispuestos en la tibia y el fémur en un mismo dispositivo que pueda combinarlos y analizarlos. Este dispositivo central, del que ya se ha hablado anteriormente, es el que contiene la pantalla, y se comunica con los otros dos dispositivos periféricos sin cables, es decir, de manera inalámbrica, para que sea más sencillo de utilizar durante la operación.

Las conexiones inalámbricas transmiten la información en forma de ondas electromagnéticas utilizando como medio de transferencia el medio aéreo. De esta manera no se requiere ningún cable u otro conector electrónico, por tanto, proporcionan mayor libertad y comodidad (Cabezas Granado, L. M., 2010).

Son varios los sistemas existentes para realizar conexiones inalámbricas. De ellos, se van a analizar a continuación los que podrían ser utilizados para el presente proyecto puesto que se han utilizado en otras ocasiones para dispositivos similares.

Comunicación por infrarrojos

La comunicación por infrarrojos envía la información mediante radiación infrarroja, que es una energía electromagnética con una longitud de onda mayor que la luz roja. Se utiliza como fuente de comunicación para dispositivos cercanos y con campo de visión directo, ya que para que sea exitosa se requiere un LED que transmita la luz infrarroja en forma de luz no visible y un fotoreceptor que la reciba (ElProCus, 2012).

Aunque este proporciona buena comunicación, no se va a poder utilizar puesto que las tres partes del dispositivo que se está diseñando no van a tener, en la mayoría de las ocasiones, campo de visión directo.

Radio

La radio fue la primera forma de comunicación sin cables utilizada. Esta permite transmitir gran cantidad de datos entre distancias cortas. Utiliza un transmisor para transmitir los datos en forma de ondas de radio a una, o varias, antenas receptoras a las que tienen que llegar por un camino libre de objetos sólidos (Mary, R., 2010).

Esta presenta el inconveniente de necesitar un camino sin objetos sólidos entre los dispositivos a comunicarse para poder transmitir datos, propiedad que no se puede garantizar

en un quirófano. Por ese motivo se descarta la utilización de la radio como método de conexión inalámbrica.

Wi-Fi

Las redes Wi–Fi utilizan ondas de radio a frecuencias no normalizadas (en el espectro libre de frecuencias) o infrarrojos para permitir a dos dispositivos comunicarse entre ellos.
Se suelen componer de una estación base con una antena de emisión y recepción, que actúa de punto de acceso creando un área de cobertura, y los dispositivos clientes, que se conectan al punto de acceso sin necesidad de cables gracias a componentes como tarjetas de red inalámbricas (Cabezas Granado, L. M., 2010).

Esta tecnología se suele utilizar para conectar rúters de internet a dispositivos como ordenadores, tabletas y teléfonos. Sin embargo, puede ser utilizado para conectar dos componentes cualesquiera, ya que además permite la conexión entre grandes distancias, aunque cabe decir que cuanto más grande sea la distancia entre los puntos de conexión, menor será la velocidad de intercambio de datos (Ray, B., 2015).

Pese a que esta sería una buena técnica gracias a el amplio espacio que puede haber entre los objetos a comunicar, presenta el problema de ser susceptible a interferencias, por lo que en este proyecto, en el que se van a utilizar sensores que pueden ser una fuente de interferencias no es una buena opción.

Bluetooth de bajo consumo (BLE)

Por último, BLE (siglas en inglés de Bluetooth de bajo consumo) utiliza ondas radio de alta frecuencia para transmitir datos y se usa para transmitir datos entre distancias cortas, con menos potencia que el Bluetooth común, por lo que suele ser utilizado para relojes inteligentes y otros dispositivos para transmitir sin utilizar mucha batería (Patkar, M., 2018).

Este solo funciona para conectar dispositivos que estén situados a una distancia corta unos de otros. No obstante, no es un inconveniente para el presente proyecto porque los dispositivos van a estar a una distancia bastante cercana, en la misma sala de quirófano.
Asimismo, no presenta interferencias de ningún tipo, por lo que es adecuado para este dispositivo.

D. <u>Lectura de datos</u>

Este dispositivo, como se ha dicho anteriormente, tiene que tener una interfaz donde leer los datos que se están calculando, es decir el ángulo relativo entre la tibia y el fémur en los tres planos anatómicos. Para ello, se va a necesitar una pantalla, y a continuación se van a explicar las características de dos tipos de ellas, las pantallas LCD y las pantallas TFT.

Pantallas LCD

La pantalla LCD está formada por píxeles colocados delante de una fuente de luz o reflectora. Aunque se dice que es una pantalla de cristal líquido, realmente se compone de cristales diminutos en un estado entre sólido y líquido (Ecured, 2018).

Pantallas TFT

Sus siglas significan transistor de pantalla delgada, y su funcionamiento está basado en transistores de efecto de campo. En este tipo de pantallas hay un electrodo sobre una placa de cristal y sobre ella capas delgadas de cristal. Cuando el electrodo activa las capas delgadas se van formando los píxeles (Informática moderna, 2019).

Ambas se podrían utilizar para este proyecto debido a sus características. Pero para este proyecto se ha utilizado una pantalla TFT.

E. Alimentación

Baterías

Como el dispositivo desarrollado ha de ser portátil, tiene que incorporar baterías para poder ser utilizado sin estar conectado a la red eléctrica.

Para ello se han utilizado baterías de polímero de ión litio, de 3,7V y 1200 mAh. Estas son recargables y en vez de utilizar un líquido como electrolito (como las típicas baterías de litio), utilizan un electrolito formado por polímero semisólido de alta conductividad (Scrosati, B., Abraham, K. M., van Schalkwijk, W. A., & Hassoun, J. (Eds.)., 2013).

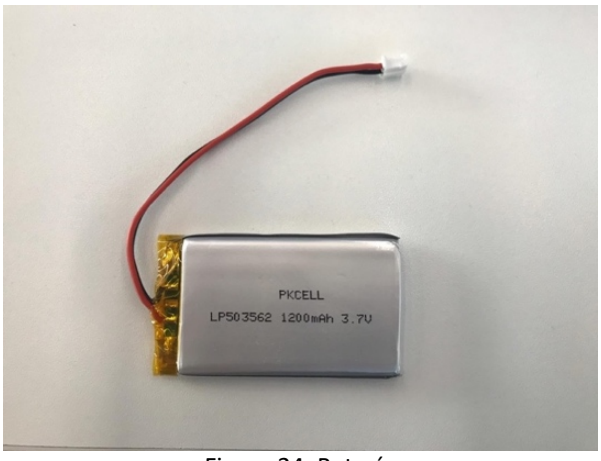


Figura 24. Batería

Como cualquier otra batería tiene cuatro componentes principales: un electrodo positivo, un electrodo negativo, un separador y electrolito. Su funcionamiento está basado en el principio

de intercalación y desintercalación de iones de litio de un material de electrodo positivo y un material de electrodo negativo con el electrolito, que proporciona el medio conductor (Scrosati, B., Abraham, K. M., van Schalkwijk, W. A., & Hassoun, J. (Eds.)., 2013).

3.2.3. Descripción de los componentes del sistema

A. Sensores

Lo primero que se estudió para el proyecto fue la manera de medir la orientación. Como se ha visto en el apartado 3.1. del presente trabajo, existen diferentes métodos para esto.

En las primeras semanas de trabajo en el proyecto se empezó a trabajar con el SiP (*System in Package*, o sistema en cápsula) flora LSM303 de la marca Adafruit, que está compuesto por un acelerómetro y un magnetómetro, ya que inicialmente se creyó que con un acelerómetro y un magnetómetro podría ser suficiente para obtener datos válidos.

Sin embargo, después de haber fusionado el código para este sensor a la placa con el microprocesador a la que se tenía que unir (Feather nRF52832), se llevo a cabo la prueba para comprobar su funcionamiento y se comprobó que este sistema sin un giroscopio no era suficientemente preciso. Aunque se introdujeran filtros manuales al código no se mejoraba la distorsión.

Por tanto, volviendo a las características estudiadas en el apartado 3.2.2. del presente trabajo, se sabía que el error encontrado en el acelerómetro era causado por las interferencias de alta frecuencia, y para reducirlo se podía utilizar junto con un giroscopio y aplicar el filtro complementario, siendo este un sistema de tracking inercial del que también se ha hablado en el apartado 3.2.2.

De esta manera, se empezó a buscar un dispositivo híbrido inercial con magnetómetro, es decir, que incorporara tres acelerómetros, tres giroscopios y tres magnetómetros, para poder obtener datos más precisos y de mejor calidad.

Entonces, se adquirió el dispositivo BNO055 de la marca Adafruit. BNO055 es un SiP que integra un acelerómetro triaxial de 14 bits, un giroscopio triaxial de 16 bit con un rango de +/- 2000 grados por segundo, y un magnetómetro de 32 bits. Los sensores son de la marca Bosch pero la placa utilizada es de la marca Adafruit, debido a que esta ya lleva incorporados los pines necesarios para conectarse al SoC (sistema en un chip)) (Sensortec, B., 2016).

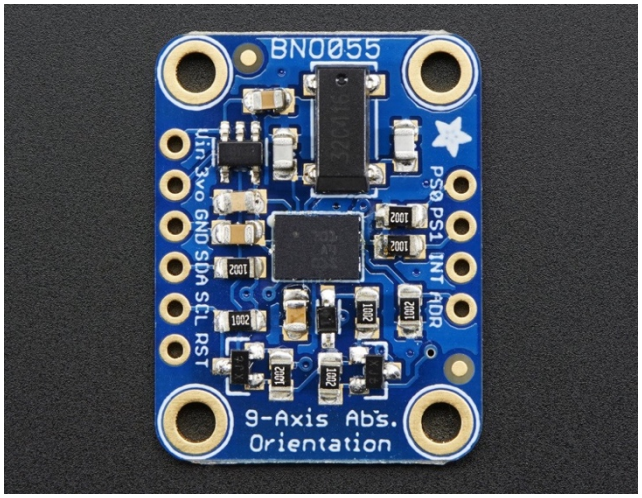


Figura 25. Sensor BNO055 (Sensortec, B., 2016).

Este SiP, tiene interfaces UART e I2C, siendo esta última la que se va a utilizar para comunicarse con el nRf52382.

En las librerías de este sensor, además, se incorporan los filtros necesarios para coger datos de los 3 sensores fusionados y evitar los errores de cada uno de los tres sensores.

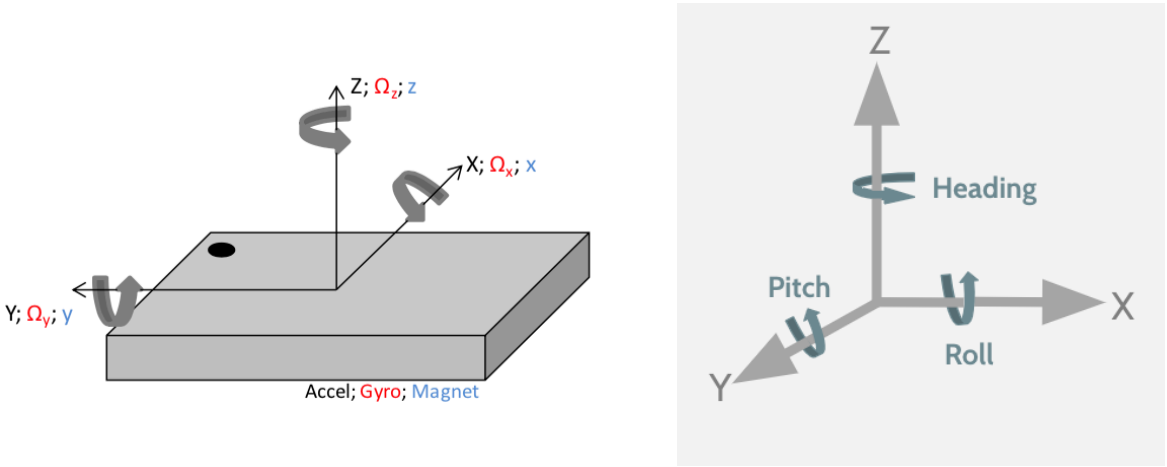


Figura 26. Ejes (Sensortec, B., 2016).

Además, el BNO055 también incorpora sensor de temperatura, por lo que si fuera necesario también se podría programar para ello. (Sensortec, B., 2016).

Los datos sin filtrar ni tratar del sensor son coordenadas X, Y y Z, que mediante los cálculos descritos en el apartado 1.2.2. para métodos híbridos de tracking, que se hacen llamando a funciones internas del sensor, se acaban transformando en ángulos.

```
X: 33.5000      Y: 2.3750       Z: -19.6250     Sys:1 G:3 A:3 M:3
X: 31.2500      Y: 1.4375       Z: -12.0000     Sys:2 G:3 A:3 M:3
X: 29.2500      Y: 3.0625       Z: -6.1875      Sys:2 G:3 A:3 M:3
X: 29.0000      Y: 5.6250       Z: -7.1875      Sys:2 G:3 A:3 M:3
X: 25.5000      Y: 1.1250       Z: -10.6875     Sys:2 G:3 A:3 M:3
X: 27.7500      Y: -3.3125      Z: -5.2500      Sys:2 G:3 A:3 M:3
X: 30.0000      Y: -6.0625      Z: -4.0000      Sys:2 G:3 A:3 M:3
X: 29.5000      Y: -3.9375      Z: -4.0000      Sys:2 G:3 A:3 M:3
X: 30.1250      Y: -2.8125      Z: -1.8750      Sys:2 G:3 A:3 M:3
X: 29.3125      Y: -1.1875      Z: -0.5625      Sys:2 G:3 A:3 M:3
X: 29.2500      Y: -0.3125      Z: -0.9375      Sys:2 G:3 A:3 M:3
X: 30.8125      Y: 1.3750       Z: -1.5625      Sys:2 G:3 A:3 M:3
X: 30.6875      Y: 2.2500       Z: -0.5000      Sys:2 G:3 A:3 M:3
X: 29.8125      Y: 1.6875       Z: -1.3125      Sys:2 G:3 A:3 M:3
X: 29.7500      Y: 1.8750       Z: -0.7500      Sys:2 G:3 A:3 M:3
X: 30.0000      Y: 1.9375       Z: 0.7500       Sys:2 G:3 A:3 M:3
X: 30.3125      Y: 2.0625       Z: 0.8125       Sys:2 G:3 A:3 M:3
X: 30.7500      Y: 2.2500       Z: 0.5625       Sys:2 G:3 A:3 M:3
X: 31.3125      Y: 2.4375       Z: 0.3750       Sys:2 G:3 A:3 M:3
X: 31.7500      Y: 2.5000       Z: 0.1875       Sys:2 G:3 A:3 M:3
X: 32.4375      Y: 2.6250       Z: 0.0625       Sys:2 G:3 A:3 M:3
X: 32.9375      Y: 2.6875       Z: 0.0000       Sys:2 G:3 A:3 M:3
X: 33.5625      Y: 2.7500       Z: -0.0625      Sys:2 G:3 A:3 M:3
X: 33.8750      Y: 2.8125       Z: -0.1250      Sys:2 G:3 A:3 M:3
X: 34.1875      Y: 2.8750       Z: -0.1875      Sys:2 G:3 A:3 M:3
X: 34.3125      Y: 2.9375       Z: -0.2500      Sys:2 G:3 A:3 M:3

Fully calibrated!
```

Figura 27. Datos en crudo del sensor y calibración.

B. <u>Microprocesador</u>

Como se ha estudiado en el apartado 3.2.2. del presente trabajo, a pesar de los diferentes métodos de conexión inalámbrica que existen actualmente, la mejor opción para este proyecto es utilizar un sistema con BLE (Bluetooth de bajo consumo), por tanto, la placa que incorpora el microprocesador para el proyecto también debe contar con conexión BLE.

Por esa y otras razones, que se explican más adelante en el documento, se adquirieron placas de la marca Adafruit, denominadas “Adafruit Feather nRF52832 Bluefruit” que incorporaban BLE.

Esta placa está compuesta por el SoC multiprotocolo de alto rendimiento (de las siglas de sistema en chip en inglés) nRF52832 de Nordic semiconductor. Desde una perspectiva de hardware, este SoC está basado en un procesador M4F con 512 kB de memoria Flash y 64 kB de memoria SRAM. Además, de los SoC que se encuentran en el mercado y que incorporan Bluetooth, este es uno de los que más capacidad tiene. Por otra parte, gestiona automáticamente la energía con un sistema propio para que su consumo sea óptimo (Nordic semiconductor, 2016).

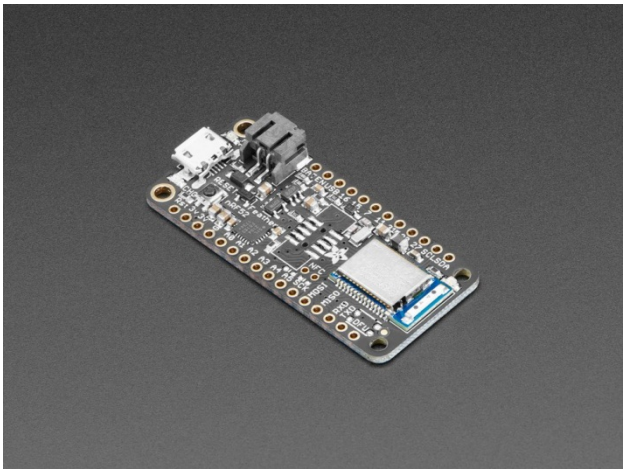


Figura 28. Placa Adafruit Feather nRF52832 (Townsend, K (2019))

Otra ventaja es que la placa Adafruit Feather nRF52832 Bluefruit, como se ve en la imagen anterior, tiene incorporados también:

- Un conector de USB
- Un conector para cargar la batería
- 19 GPIO
- Un LED azul que parpadea cuando la placa está encendida
- Un LED rojo, que parpadea para los hechos que se haya programado.
- Un botón de reinicio

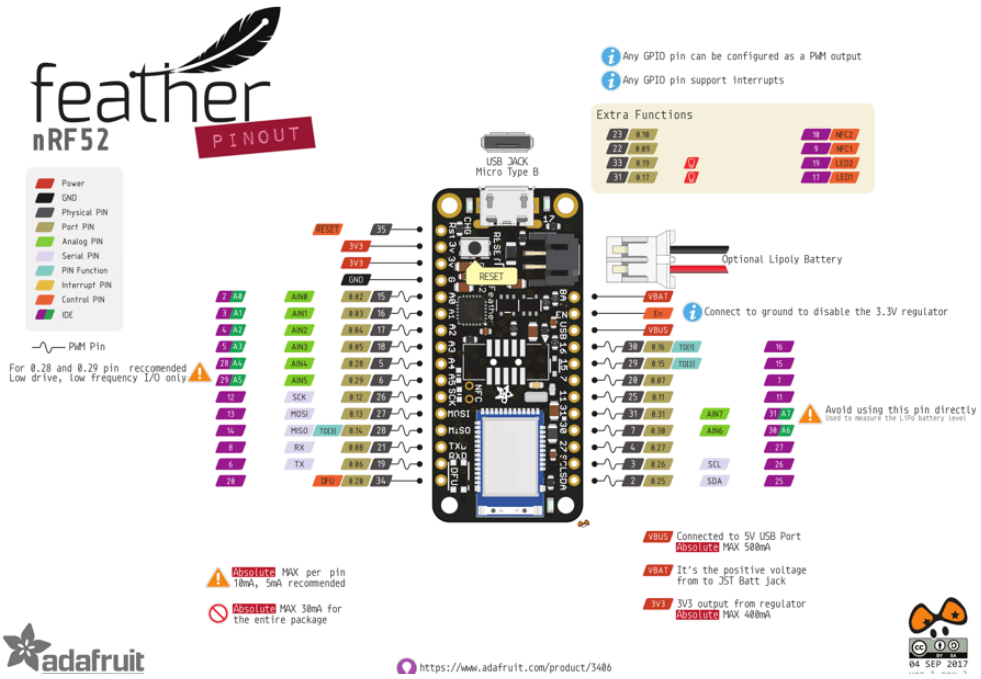


Figura 29. Disposición de los pines en la placa (Townsend, K, 2019).

Y, por último, otra de sus características es que se puede programar directamente desde el Arduino IDE, por lo que se hace más fácil su programación.

Para este proyecto, se utilizaron tres placas de este tipo. Dos que hacen función de dispositivos periféricos, que se encuentran posicionados en los huesos tibia y fémur, y otro que hace función de dispositivo central, colocado junto a una pantalla que hará la función de mostrar los datos al cirujano.

Asimismo, este SoC tiene interfaz SPI y I2C, por lo que se puede conectar a otros dispositivos (en este caso la pantalla TFT y los sensores BNO055) con cualquiera de las dos interfaces de comunicación (Townsend, K., 2019).

C. Pantalla

El dispositivo central del sistema consiste en una placa Feather nRF52832, de las que se han descrito anteriormente, conectada a una pantalla TFT Feather Wing de 3,5 pulgadas, con 6 LED blancos, de 480x320 píxeles con un control individual de píxeles de color de 16 bits. Esta es una pantalla táctil resistiva que puede detectar pulsaciones en cualquier parte de la pantalla (Ada, L., 2019).

Utiliza interfaz SPI para recibir datos desde el nRF52832 (Ada, L., 2019).

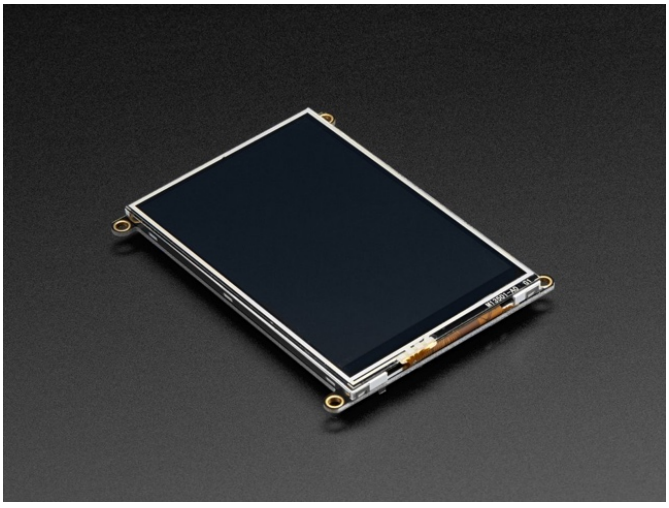


Figura 30. Pantalla TFT Feather Wing (Ada, L, 2019).

D. Envoltura física

Para que las placas y conexiones del dispositivo no queden al aire completamente, pudiendo quebrarse en cualquier momento, se ha diseñado una envoltura física para introducirlo. Para ello se ha hecho diseño 3D en el programa Fusion 360.

Fusion 360 es un software de Autodesk que sirve para el diseño de productos. Este programa tiene una sencilla utilización, lo que permite el diseño de productos en 3D para después imprimirlos en cualquier impresora 3D.

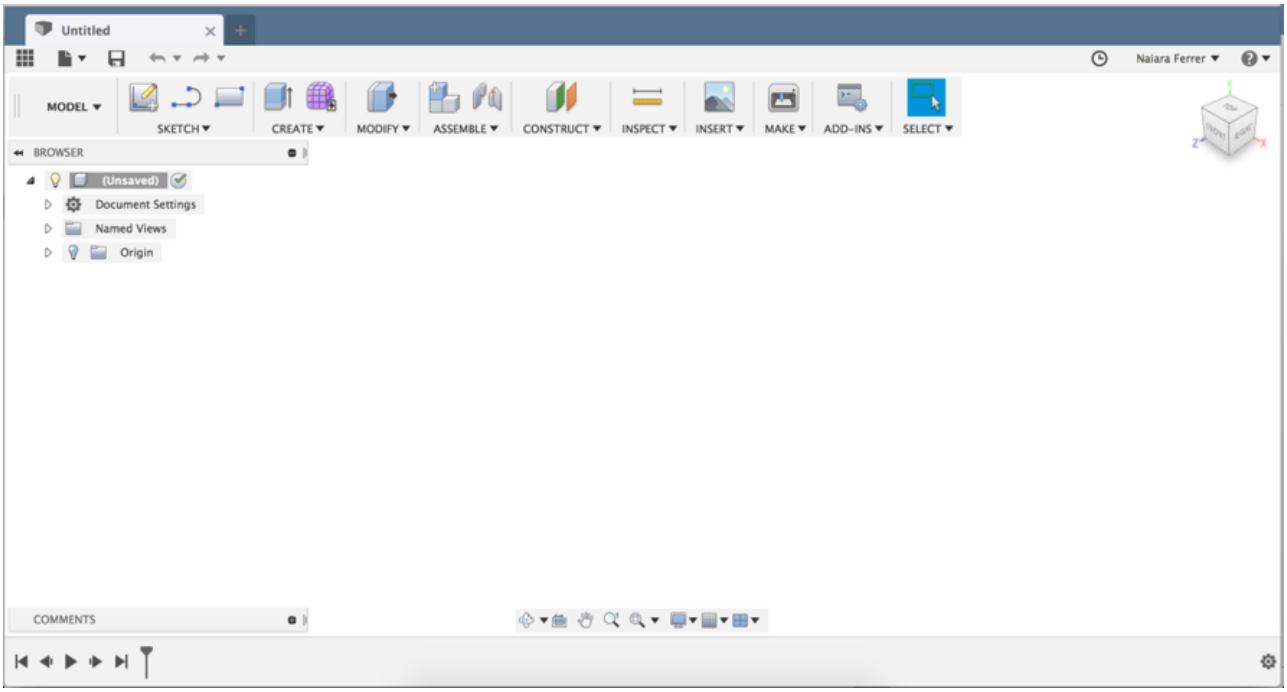


Figura 31. Software Fusion 360 de Autodesk

Una vez realizado el diseño se ha introducido en Ultimaker cura. Este es un software de impresión 3D que permite utilizar los archivos creados con Fusion 360 u otros programas de diseño y prepararlos directamente para enviarlos a una impresora 3D, seleccionando el tipo de material a utilizar, el número de capas de material, la velocidad, etc.

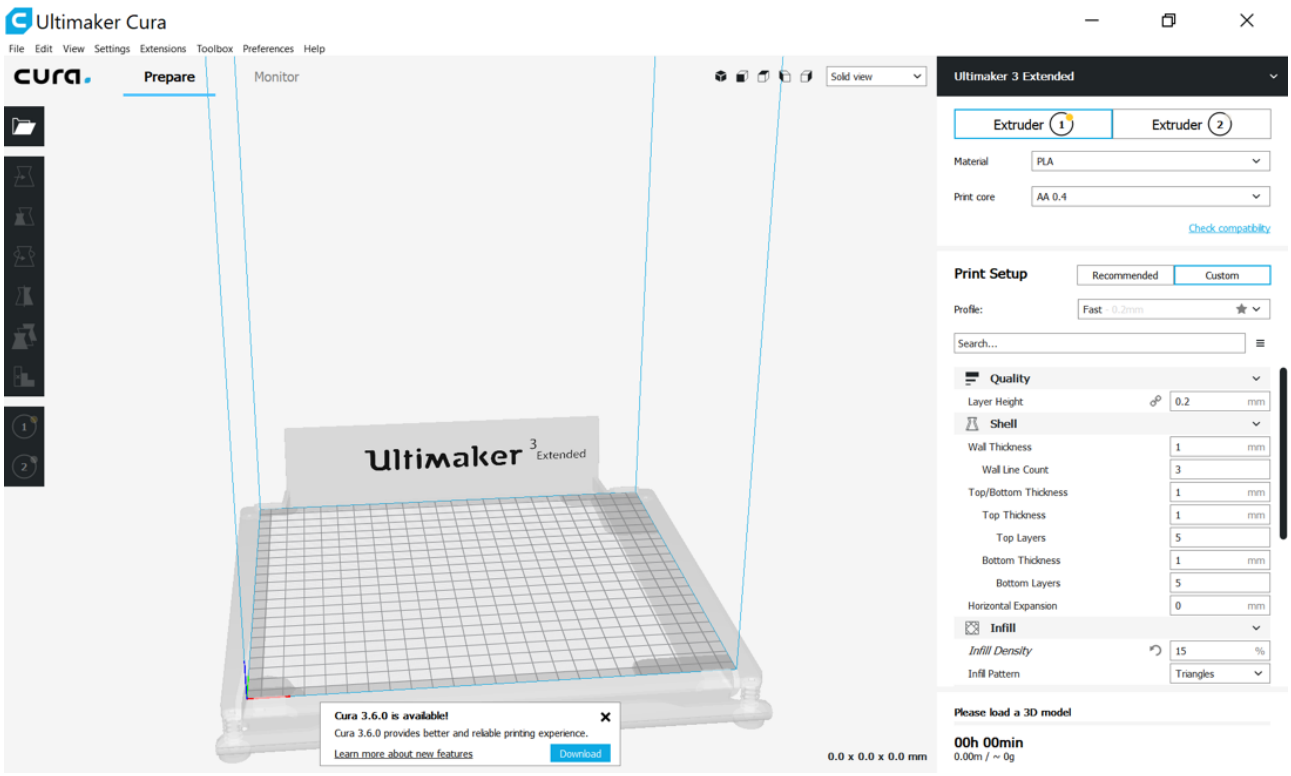


Figura 32. Software Ultimaker Cura

Por último, con la impresora Ultimaker 3D, que es una impresora 3D que combina materiales de soporte para la construcción y solubles en agua para crear piezas, se ha llevado a cabo la impresión de las distintas cajas para el dispositivo.

Figura 33. Impresora 3D Ultimaker.

A continuación, se muestran las imágenes de las envolturas obtenidas.

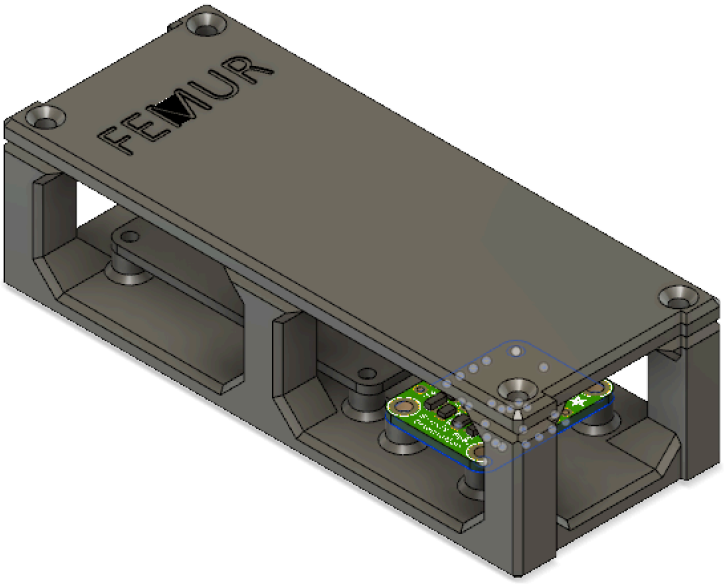


Figura 34. Diseño de la envoltura para el dispositivo femoral, con las mismas características que el tibial.

Figura 35. Diseño de la envoltura para el dispositivo femoral, con las mismas características que el tibial. Visión lateral

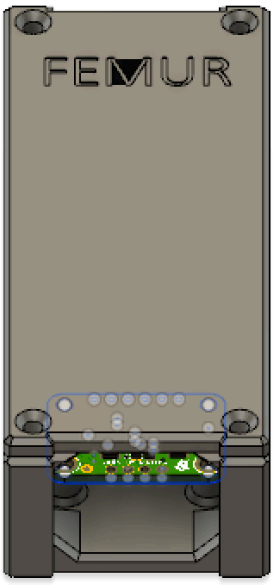


Figura 36. Diseño de la envoltura para el dispositivo femoral, con las mismas características que el tibial.

Figura 37. Envoltura para los dispositivos periféricos (sin las tapas).

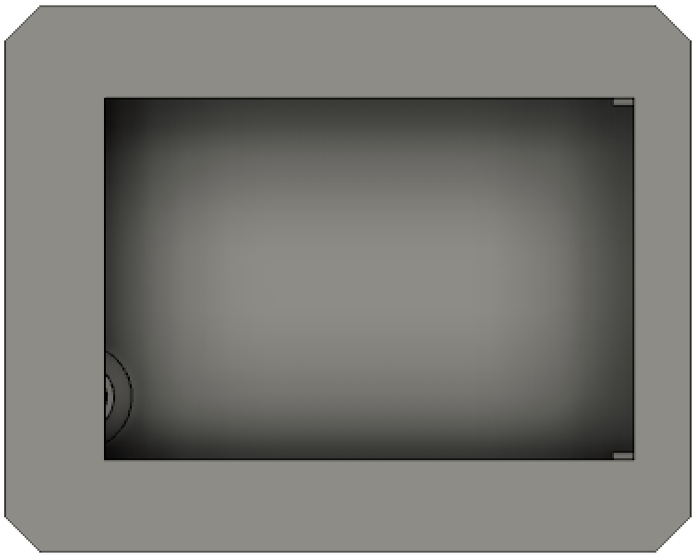


Figura 38. Diseño de la envoltura para el dispositivo central, con hueco para la pantalla

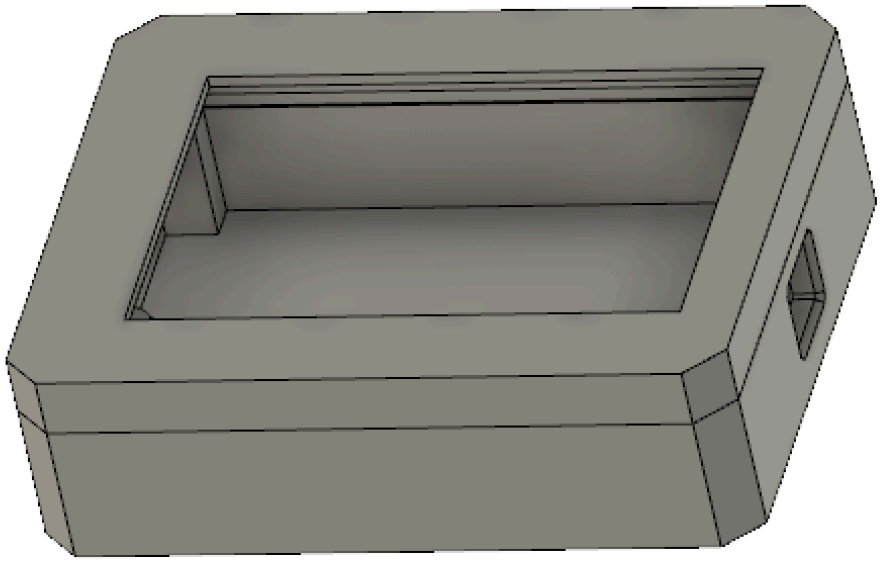


Figura 39. Diseño de la envoltura para el dispositivo central.

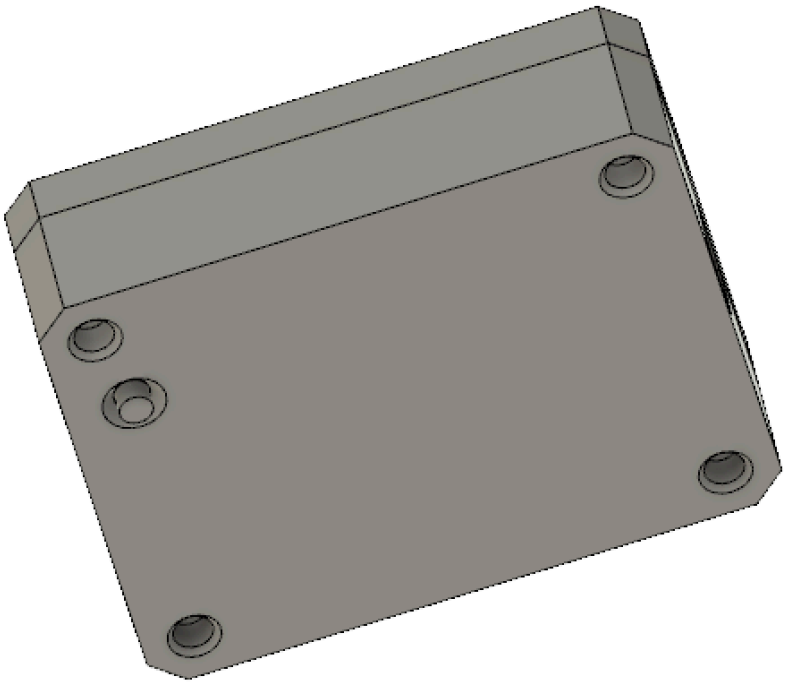


Figura 40. Diseño de la envoltura para el dispositivo central, con botón trasero para reiniciar la pantalla.

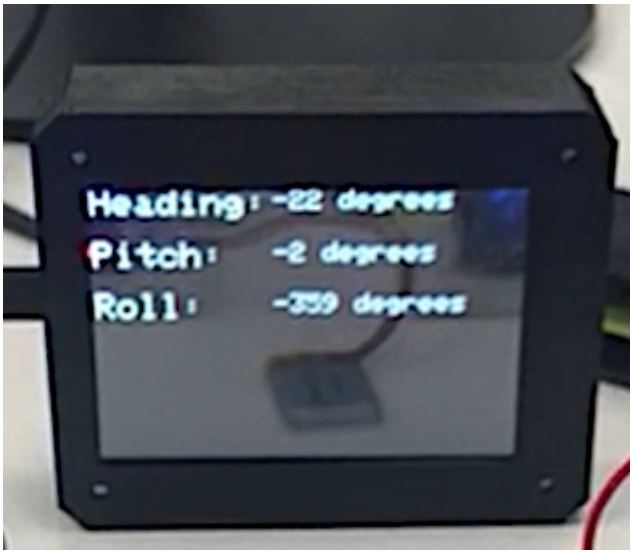


Figura 41. Envoltura para el dispositivo central, con la pantalla incorporada.

3.3. SOFTWARE / FIRMWARE DEL MICROCONTROLADOR

El firmware es definido por el Instituto de Ingenieros Eléctricos (IEEE) como "La combinación de instrucciones de un dispositivo de hardware e instrucciones y datos de computadora que residen como software de solo lectura en ese dispositivo". Es decir, es la lógica de más bajo nivel que se utiliza para controlar un dispositivo electrónico mediante sus circuitos electrónicos. Por tanto, es un software, pero su función básica es controlar el hardware físicamente y se almacena en la memoria ROM, que es la memoria de solo lectura, para funcionar cada vez que se encienda la placa (Doña, D.).

A continuación, se van a explicar las diferentes partes del firmware y el código implementado.

3.3.1. Interfaces de comunicación

A. I2C

El primer sistema de comunicación es la interfaz I2C, que se utiliza para la comunicación entre el sensor y el microcontrolador. Este es un sistema de comunicación entre memorias, microcontroladores, etc. que solo requiere dos líneas de señal bidireccional. Su metodología de comunicación es en serie y sincrónica, ya que una de las señales se utiliza para intercambiar datos y la otra para marcar el tiempo. Su velocidad está entre los 100 Kbits/segundo y 3,4 MHz (Carletti, E. J. (2007); Osio, J. R., Antonini, L., Aróztegui, W., & Rapallini, J. A., 2011).

Está compuesto por 3 conectores:

- SCL, *system clock*, que como su nombre indica es el que se encarga de los pulsos de reloj para sincronizar el sistema
- SDA, *system data*, el encargado de intercambiar datos
- GND, masa, que es el punto común entre todos los dispositivos conectados al sistema.

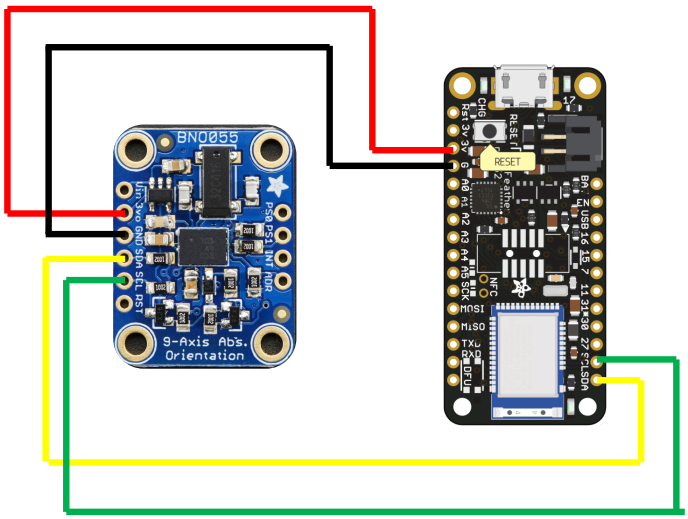


Figura 42. Conexión entre el sensor y la placa Arduino.

Se pueden conectar hasta 128 dispositivos, pero existe un protocolo en el que los dispositivos tienen que ser maestros o esclavos, y solo los maestros pueden iniciar una comunicación.

Cuando el bus está libre, es decir, tanto la línea SCL como la línea SDA están inactivas, cualquier maestro puede ocuparlo y se establece la condición de *start*, entonces se transmite un byte con siete bits correspondientes a la dirección única que tiene asignada el dispositivo esclavo que quiere seleccionar, y otro bit que indica si se quiere realizar una operación de lectura o de escritura (R/W).

Si se encuentra el esclavo correspondiente a la dirección enviada se envía un bit de reconocimiento (ACK) para establecer la comunicación e intercambiar información.

Si el bit que se ha enviado desde el maestro es de escritura cuando se establezca la comunicación se enviarán todos los datos mientras se reciban señales ACK o hasta que se envíen todos. Siempre se empieza enviando el bit más significativo (MSB) (Carletti, E. J. (2007); Osio, J. R., Antonini, L., Aróztegui, W., & Rapallini, J. A., 2011).

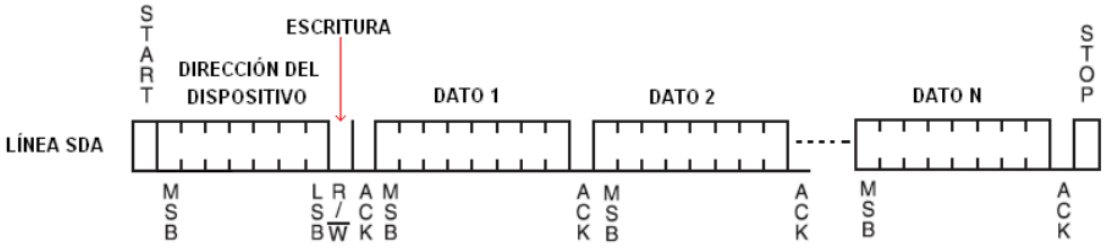


Figura 43. Funcionamiento de la interfaz I2C (Osio, J. R., Antonini, L., Aróztegui, W., & Rapallini, J. A. (2011)).

Si el octavo bit es de lectura, lo que ocurre es que el SCL del maestro genera pulsos de reloj y entonces el esclavo envía los datos. En este caso, el ACK lo envía el maestro, después de cada byte recibido, hasta que el maestro genere una parada (*STOP*) (Carletti, E. J. (2007); Osio, J. R., Antonini, L., Aróztegui, W., & Rapallini, J. A., 2011).

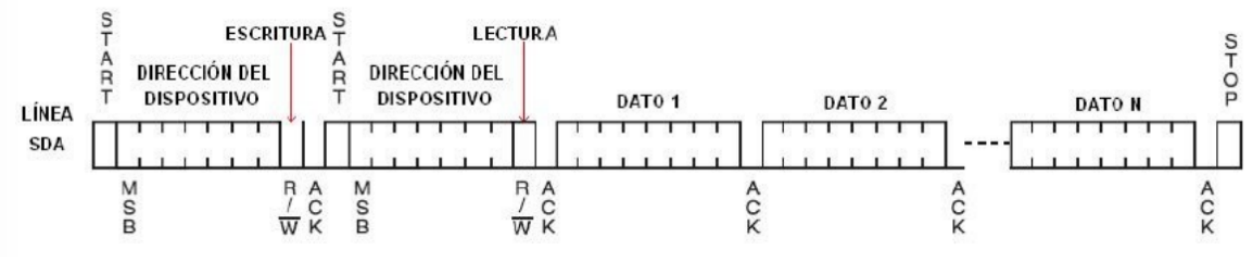


Figura 44. Funcionamiento de la interfaz I2C (Osio, J. R., Antonini, L., Aróztegui, W., & Rapallini, J. A. (2011)).

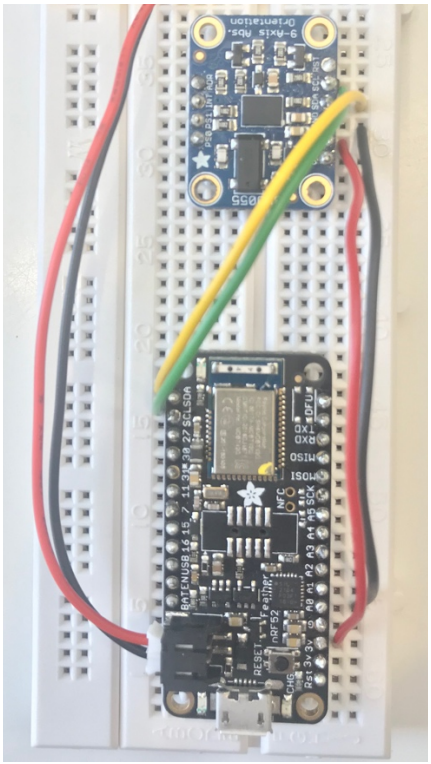


Figura 45. Conexión de la placa Bluefruit nRF52 con el sensor BNO055 mediante I2C.

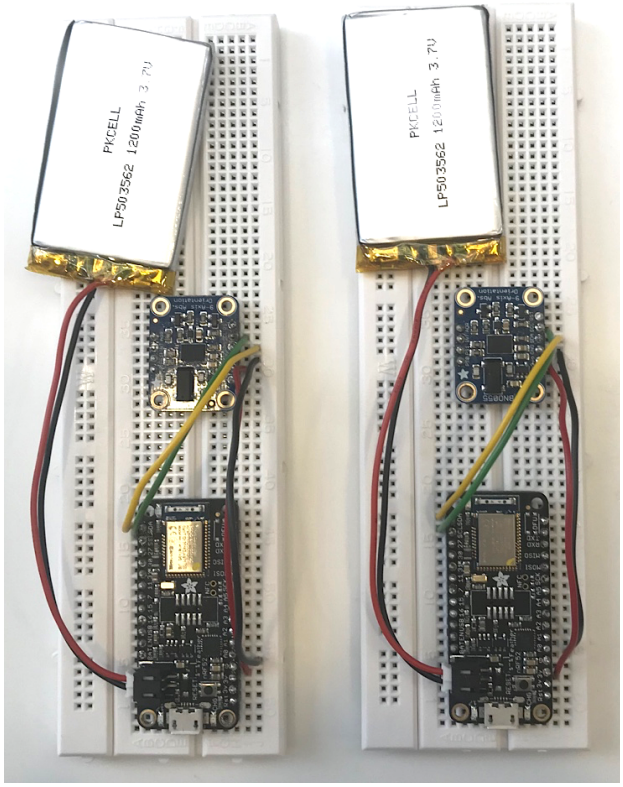


Figura 46. Conexión de la placa Bluefruit nRF52 con el sensor BNO055 mediante I2C.

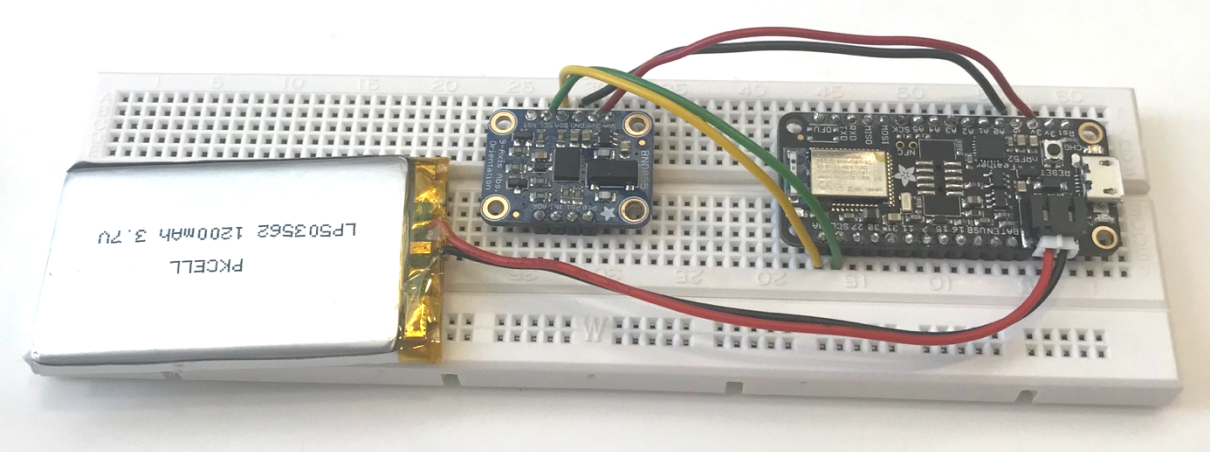


Figura 47. Conexión de la placa Bluefruit nRF52 con el sensor BNO055 mediante I2C.

B. <u>SPI</u>

La pantalla y el microcontrolador del dispositivo central se comunican con la interfaz SPI (*Serial peripheral interface*). Es una interfaz sincrónica máster – esclavo basada en un registro de desplazamiento de 8 bits. Es decir, existe un dispositivo, que es el mater, que se encarga de generar la sincronía, y uno o más esclavos. Al final el registro es de 16 bits. Está compuesta por 4 conectores:

- SS, selector de esclavo, que se encarga de seleccionar el dispositivo con el que se va a comunicar.
- SCLCK, *serial clock*, que genera una señal que es utilizada por todos los dispositivos para coordinar la transferencia de datos.
- MOSI, *master output / slave input*
- MISO, *master input / slave output*

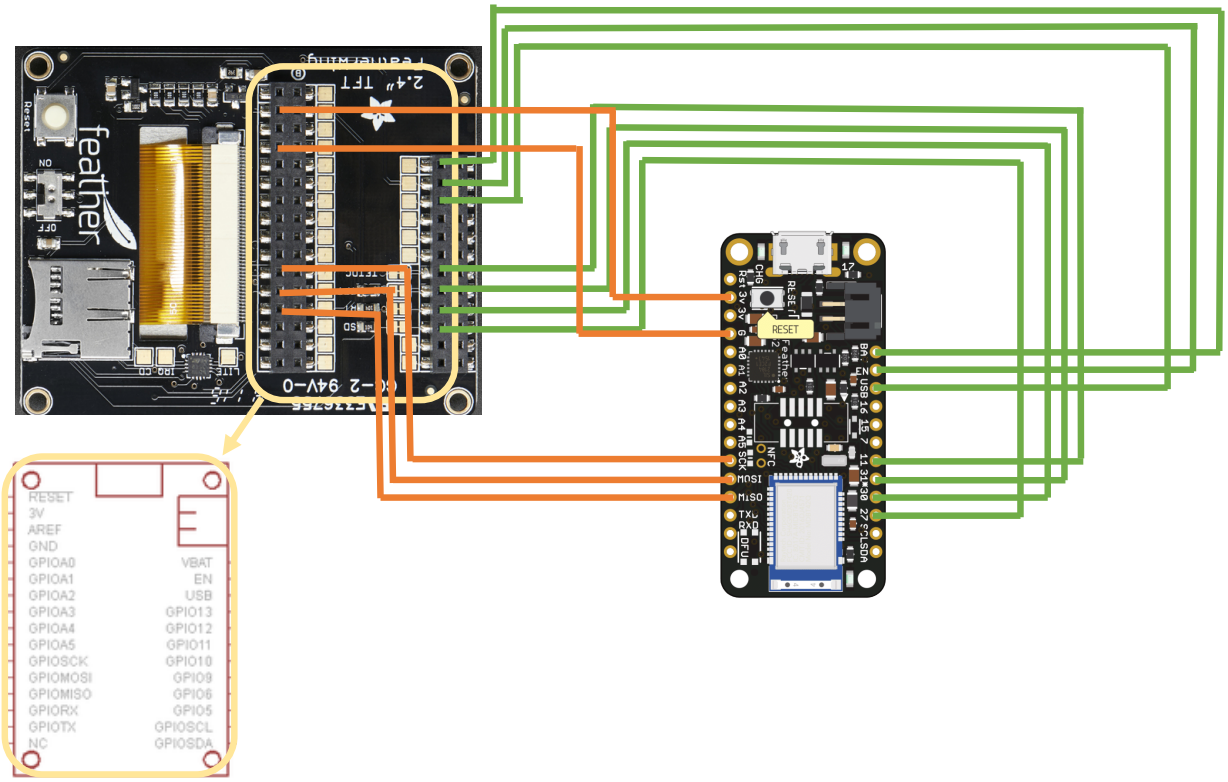


Figura 48. Conexión entre la pantalla y la placa Arduino

La comunicación SPI empieza cuando el SCLK genera un pulso. En este momento el último bit se manda por el MOSI si es el máster, o por el MISO si es el esclavo, y el último bit del otro dispositivo se recibe a través del MOSI si es el máster o del MISO si es el esclavo, y entonces se almacena en el primer puesto del registro, desplazando los demás bits una posición.

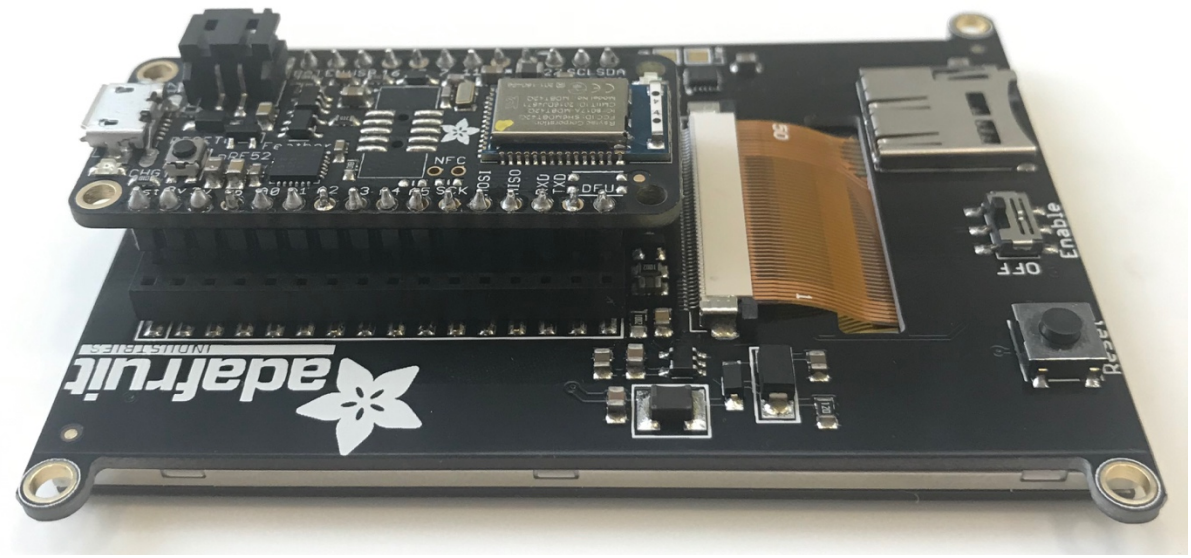


Figura 49. Conexión de la placa Bluefruit nRF52 con la pantalla TFT FeatherWing mediante SPI.

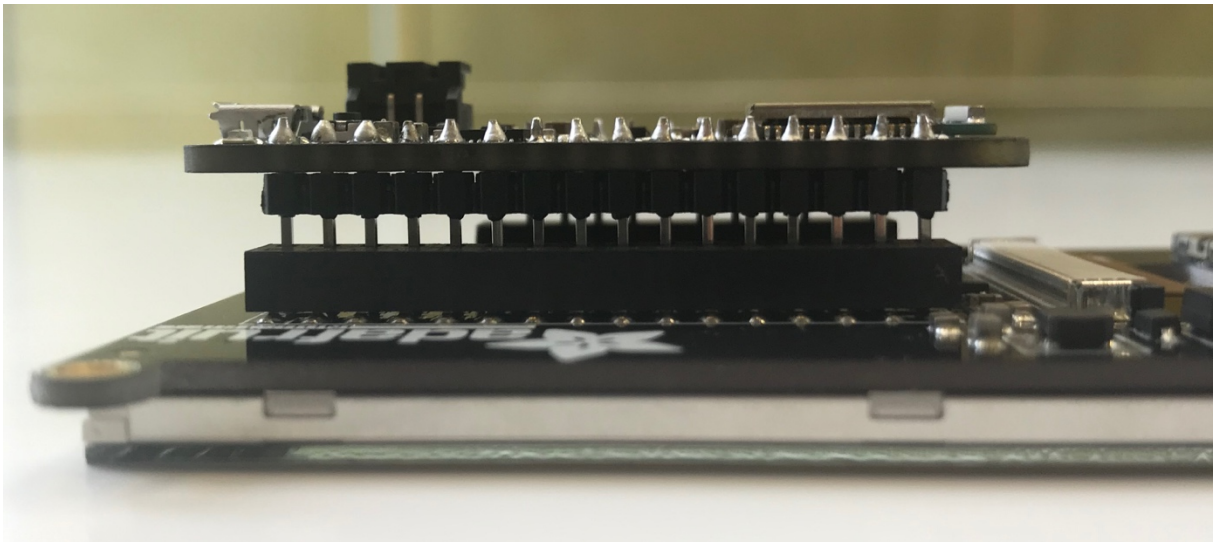


Figura 50. Conexión de la placa Bluefruit nRF52 con la pantalla TFT FeatherWing mediante SPI.

C. BLE UART

El segundo sistema de comunicación es el BLE UART, que es el que utilizan para comunicarse entre ellos los diversos microcontroladores del sistema.

El UART (*Universal Asynchronous Receiver Transmitter*) es la interfaz más popular para la transmisión de datos. En este tipo de comunicación no hay señal de pulsos de reloj, por tanto, los dispositivos que se comuniquen, es decir, el transmisor y el receptor, tienen que llevar la misma velocidad de bits por segundo para que no haya pérdida de información.

Este estándar especifica que todo lo que se transmite va precedido de un bit de inicio (*start),* cuya única función es indicar que llega un nuevo dato, ya que no contiene información. Una vez se han recibido los datos, estos van seguidos de uno o más bits de parada (*stop*).

Los datos se envían agrupados en bytes, o lo que es lo mismo, 8 bits. Estos se colocan en el bus de datos del sistema del dispositivo emisor, a continuación, se transfieren a un registro de desplazamiento de este, y este transmite cada bit en serie al dispositivo periférico. Como siempre se reciben agrupados en 8 bits, el receptor puede detectar cuando ha terminado de recibir datos (Michael, M. S., 1992).

En este caso, no se utiliza el UART tradicional, sino una emulación de puerto serie sobre BLE, llamada BLE UART, que se está incorporando en prácticamente todas las placas que contienen BLE.

En los sistemas clásicos de Bluetooth, un perfil de puerto serie (SPP) es un perfil adoptado definido por el grupo de interés especial (SIG) de Bluetooth para emular una conexión de

puerto serie a través de una conexión inalámbrica de Bluetooth. Pero para los sistemas de Bluetooth de bajo consumo no está definido ningún perfil de puerto serie, por lo tanto, en muchas ocasiones se utiliza el BLE UART como equivalente (Instruments, T, 2015).

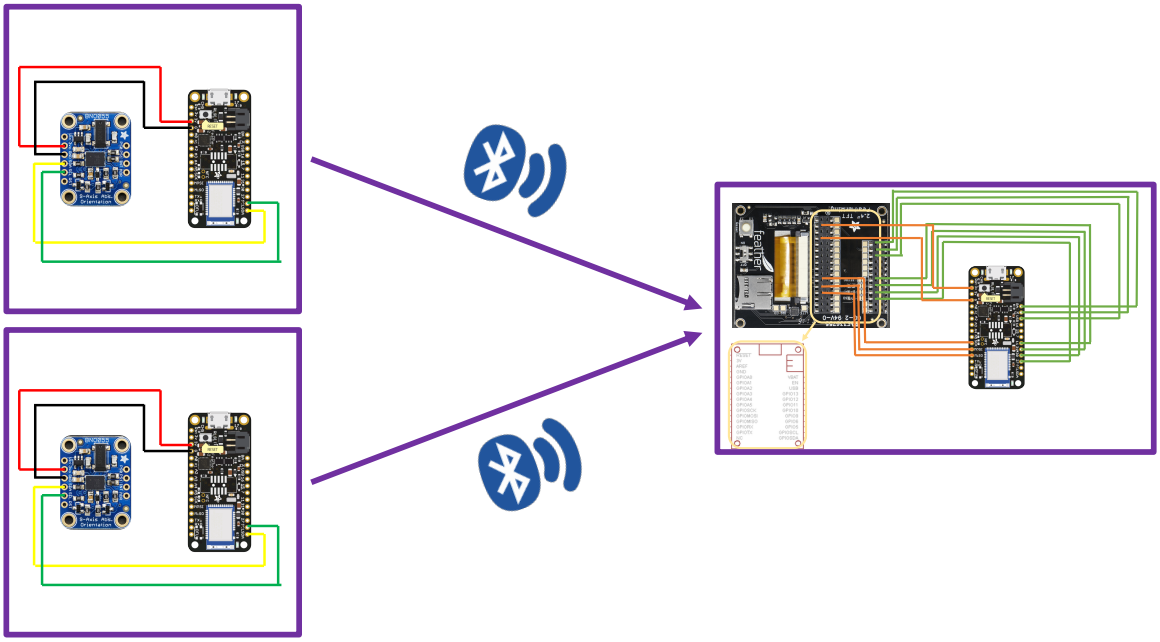


Figura 51. Conexión entre los dispositivos periféricos y el dispositivo central.

En esta interfaz los dispositivos adquieren la condición de servidor o cliente, y cada byte puede ser enviado y recibido tanto por el servidor como por el cliente.

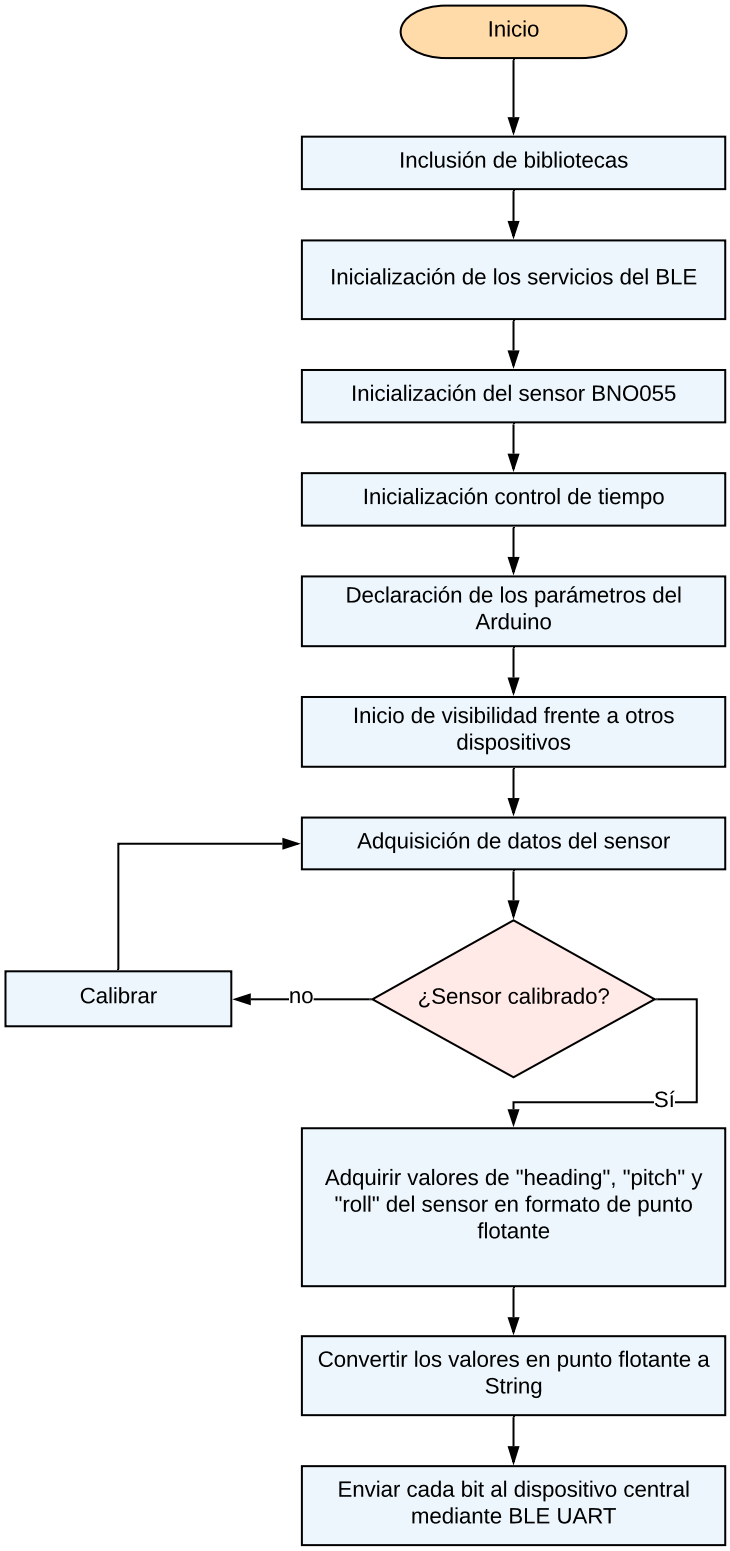
3.3.2. Código

3.3.2.1. Diagrama de flujo

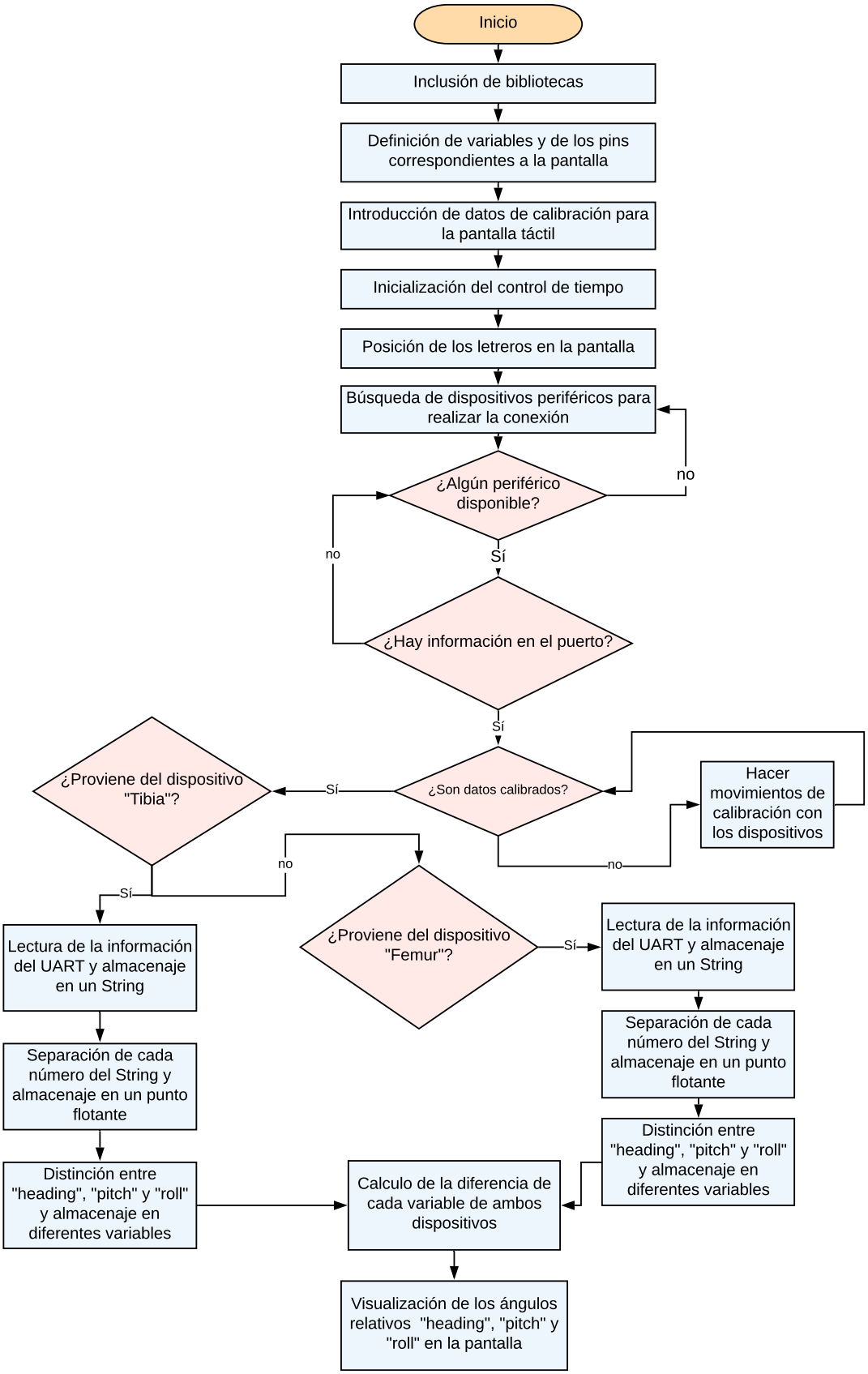
Se va a emplear un diagrama de flujo para cada uno de los códigos para poder llevar a cabo la explicación de su funcionamiento. El código, en lenguaje C++, se puede consultar en el Anexo 1.

Este diagrama de flujo es la representación gráfica del código mediante bloques de acción, representados por rectángulos, y bloques de decisión, representados por rombos. Con él se pretende facilitar la comprensión del código evitando el enredo que puede producir la lectura del código.

El primer diagrama de flujo corresponde al código para los sensores periféricos, es decir, los sensores que se colocarán en los huesos tibia y fémur.



A continuación, se muestra el diagrama de flujo del código del dispositivo central, es decir, el dispositivo que recibe los datos y lleva incorporada la pantalla.


Inicio
Inclusión de bibliotecas
Definición de variables y de los pins correspondientes a la pantalla
Introducción de datos de calibración para la pantalla táctil
Inicialización del control de tiempo
Posición de los letreros en la pantalla
Búsqueda de dispositivos periféricos para realizar la conexión
¿Algún periférico disponible?
no
Sí
no
¿Hay información en el puerto?
Sí
¿Son datos calibrados?
Sí
¿Proviene del dispositivo "Tibia"?
no
Hacer movimientos de calibración con los dispositivos
no
Sí
Lectura de la información del UART y almacenaje en un String
Separación de cada número del String y almacenaje en un punto flotante
Distinción entre "heading", "pitch" y "roll" y almacenaje en diferentes variables
¿Proviene del dispositivo "Femur"?
Sí
Lectura de la información del UART y almacenaje en un String
Separación de cada número del String y almacenaje en un punto flotante
Distinción entre "heading", "pitch" y "roll" y almacenaje en diferentes variables
Calculo de la diferencia de cada variable de ambos dispositivos
Visualización de los ángulos relativos "heading", "pitch" y "roll" en la pantalla

3.3.2.2. Aspectos destacables del código

El funcionamiento del programa se puede entender con los diagramas explicados en el apartado anterior. No obstante, también es fundamental explicar con más detalle algunos aspectos del código para comprender mejor como funciona el sistema.

A. Descarga de bibliotecas

Para poder utilizar los sensores y placas con un código creado en el software Arduino IDE es necesario la descarga de ciertas bibliotecas que aportan información extra o funciones internas, y se deben incluir en los códigos. Estas bibliotecas son las siguientes:

- Bluefruit: Contiene las funciones internas de la placa de Arduino Bluefruit nRF52832
- Adafruit_Sensor y Adafruit_BNO055: Contienen las funciones internas de los sensores de los dispositivos periféricos, por tanto, solo hay que añadirlas en los códigos correspondientes a estos.
- Wire: Es necesaria para poder realizar la conexión I2C de la que se ha hablado anteriormente.
- Utility/imumaths: Sirve para realizar operaciones internas como cambio del sistema de médida.
- Adafruit_GFX: Aporta una sintaxis y conjunto de funciones gráficas para utilizar en pantallas de la marca Adafruit. Esta solo se debe incluir en el código del dispositivo central, que es el que contiene la pantalla.
- Adafruit_HX8357: Contiene funciones internas de la pantalla que se ha utilizado en el proyecto, por tanto, también se incluirá solo en el código del dispositivo central.
- TouchScreen: Aporta los elementos necesarios para la utilización de la pantalla táctil.
- SPI: Necesaria para realizar la conexión SPI, ya explicada anteriormente.
- Adafruit_STMPE610: Controla los elementos resistivos de la pantalla táctil

```
#include <Adafruit_GFX.h>
#include "Adafruit_HX8357.h"
#include <bluefruit.h>
#include "TouchScreen.h"
#include <SPI.h>
#include <Adafruit_STMPE610.h>
```

Figura 51. Bibliotecas incluidas en el código del dispositivo central.

```
#include <bluefruit.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>
```

Figura 53. Bibliotecas incluidas en el código de los dispositivos periféricos.

B. Envío de datos

En el sistema BLE UART la comunicación se lleva a cabo mediante bytes (8 bits), por tanto, los datos que se manden desde los dispositivos periféricos al dispositivo central para llevar a cabo las operaciones necesarias se tienen que mandar en dicho formato.

En la imagen siguiente se puede ver la parte del código en la que se hace la conversión.

```
delay(100);
float heading = euler.x(); // Valor en punto flotante


sheading = String(heading); // Convertido en String
uint8_t utf[sizeof(sheading)];

// Separación en bits
for(int l=0; l <sizeof(sheading)+1; l++){
  utf[l] = sheading.charAt(l);
}

bleuart.write( utf, sizeof(sheading)); // Envío de cada bit
```

Figura 54. Parte del código del dispositivo periférico en la que se realiza la conversión que permite enviar datos al dispositivo central.

Para empezar, el sensor BNO055 recoge los datos en forma de *float* (que son 4 bytes), por tanto, así es como lo recibe la placa nRF52832 cuando llama a la función Euler.x().

Este valor, a continuación, se almacena en un *String*, con nombre sheading, y se crea una variable, a la que se ha llamado utf, con formato uint8_t (número entero positivo de 1 byte) del tamaño del *String* que contiene los datos del sensor, es decir, la posición angular en la que se encuentra el dispositivo.

Una vez se ha creado la variable utf, se implementa un bucle “for” para colocar dentro de esta variable los datos del *String* anterior sheading. Entonces los datos ya están listos para mandarse mediante comunicación BLE UART al dispositivo central.

El proceso realizado anteriormente se vuelve a llevar a cabo para los valores de pitch y roll.

C. Recepción de datos

El primer trabajo que hacer con los datos recibidos es identificar si se han recibido desde el dispositivo femoral o desde el dispositivo tibial.

```
while ( uart_svc.available() )
{
  char buf[20 + 1] = { 0 };
  if ( uart_svc.read(buf, sizeof(buf) - 1) )
  {
     // Si la información viene de la tibia, se recogen los datos en un String
    if (strcmp(peer->name, "Tibia sensor") == 0) {
      for (int i = 0; i < 3; i++) {
        tibia[i] = String (buf);
      }

      // Se separa cada uno de los números en un punto flotante
      float tib1 = tibia[0].toFloat();
      float tib2 = tibia[1].toFloat();
      float tib3 = tibia[2].toFloat();
```

Figura 55. Parte del código del dispositivo central en la que se realiza la conversión de los datos recibidos desde el dispositivo tibial.

Por tanto, como se ve en la imagen anterior, se implementa un bucle “if” que especifica que si el nombre del dispositivo que envía los datos es “Tibia sensor”, que es el nombre que se le ha puesto al dispositivo de la tibia cuando se ha inicializado su código, coja los datos que se encuentran en el búfer, es decir, donde se colocan los datos recibidos, y los ponga dentro de una variable de formato *String* llamada tibia. Como los datos llegan en formato de 1 byte, se recogen en ese *String* pero después se transforman cada uno de ellos en *float* con el comando “.toFloat()”. Así lo que quedan son tres variables de formato *float* tib1, tib2 y tib3, que corresponden a cada uno de los tres datos recibidos desde el dispositivo tibial, es decir, corresponden al *heading, pitch* y *roll*, del dispositivo colocado en la tibia.

El procedimiento para los datos recibidos desde el sensor femoral, al que se le ha llamado “Femur sensor”, es el mismo que para la tibia, como se puede ver en la siguiente imagen.

```
if (strcmp(peer->name, "Femur sensor") == 0) {
  for (int j = 0; j < 3; j++) {
    femur[j] = String (buf);

  }
  float fem1 = femur[0].toFloat();
  float fem2 = femur[1].toFloat();
  float fem3 = femur[2].toFloat();
```

Figura 56. Parte del código del dispositivo central en la que se realiza la conversión de los datos recibidos desde el dispositivo femoral.

La única diferencia es que en este caso el *string* donde se van recogiendo los datos es nombrado "femur" y que las variables donde se recogen el *heading, pitch*, y *roll,* se llaman fem1, fem2 y fem3.

Sin embargo, hay que tener en cuenta que fem1/tib1, fem2/tib2 y fem3/tib3 no son el *heading*, pitch y roll en ese orden, sino que se van recibiendo conforme se van adquiriendo y en este punto no se sabe aún cual es cual. Por tanto, habrá que implementar otra parte de código para distinguir cada uno de los datos recibidos por ambos sensores.

D. Distinción entre *heading, pitch* y *roll*

En esta sección se va a comentar como se hace la distinción de los tres datos que se requieren en el código, teniendo en cuenta que los 3 se encuentran en el mismo intervalo de números.

Los datos que adquieren los sensores situados en los huesos del fémur y la tibia son los ángulos en los que se encuentran en los tres ejes anatómicos, es decir, *heading, pitch* y *roll.* Esto quiere decir que los tres datos se encuentran en un intervalo comprendido entre el -360 y el 360.

Cuando el dispositivo central recibe los datos que proceden de ambos dispositivos periféricos distingue de cual de los dos dispositivos proviene, es decir, identifica si los datos son del fémur o de la tibia. Sin embargo, desde los dispositivos periféricos no se manda una señal para determinar si el dato enviado corresponde al *heading*, al *pitch*, o al *roll.* Por tanto, esta clasificación se tiene que hacer en el dispositivo central una vez se han recibido todos los datos.

Para poder llevar a cabo la distinción, la primera parte de la operación se encuentra en el código de los dispositivos periféricos.

```
  // ------------------------------------------------ PITCH ----------------------------------------------------//

  float pitch = euler.z();

  pitch = pitch + 1000; // Se suma 1000 para tener una distinción entre las otras  dos variables
  spitch = String(pitch);
  uint8_t vtf[sizeof(spitch)];

  for(int k=0;k<sizeof(spitch)+1; k++){
   vtf[k]=spitch.charAt(k);
  }

  bleuart.write(vtf, sizeof(spitch));

// ------------------------------------------------ ROLL -----------------------------------------------------//

  float roll = euler.y();
  if (roll < 0){
    roll=roll+360;
  }
  roll = roll + 2000; // Se suma 2000 para hacer distinción entre las variables

sroll = String(roll);
uint8_t wtf[sizeof(sroll)];

for(int j=0;j<sizeof(sroll+1); j++){
  wtf[j]=sroll.charAt(j);
}

bleuart.write(wtf, sizeof(sroll));
```

Figura 57. Parte del código de los dispositivos periféricos en los que se obtienen los datos de pitch y roll y se hace la distinción necesaria.

Como se puede observar en el código anterior, en la parte del ángulo correspondiente al *pitch* se ha sumado 1000 a su valor natural, y al ángulo correspondiente al *roll* se le ha sumado 2000. Esto es para que, ahora, los datos del *heading* estén comprendidos entre el 0 y el 360, los datos del *pitch* entre el 820 y el 1180 y los datos del *roll* entre el 2000 y el 2360.

La segunda parte de la operación para distinguir cada uno de los datos está en el código del dispositivo central.

```
if (fem1 < 500) {
  hfemur = fem1;
}

if (fem1 > 815 && fem1 < 1200) {
  pfemur = fem1;
}

if ( fem1 > 1999){
  rfemur = fem1;
}

if (fem2 < 500) {
  hfemur = fem2;
}

if (fem2 > 815 && fem2 < 1200) {
  pfemur = fem2;
}

if ( fem2 > 1999){
  rfemur = fem2;
     }

if (fem3 < 500) {
  hfemur = fem3;
}

if (fem3 > 815 && fem3 < 1200) {
  pfemur = fem3;

 if ( fem3 > 1999){
    rfemur = fem3;
       }
```

Figura 58. Parte del código del dispositivo central donde se hace la distinción entre los diferentes datos recibidos.

Una vez los datos recibidos se han convertido al formato adecuado para poder leerse se tienen que clasificar en *heading, pitch* o *roll* para poder realizar las operaciones necesarias con cada uno de ellos y mostrarlos en la pantalla.

En la parte del código que se ve en la figura 58, se puede observar que para distinguir cada uno de los 3 datos recibidos de cada sensor se implementan diferentes bucles “if”.

Como en el código de los periféricos se había sumado 1000 al valor del *pitch* y 2000 al valor del *roll,* lo que se ha hecho aquí es decirle al programa que si tib1 (que como se ha visto anteriormente, en el apartado de “Recepción de datos”, es el primer dato que entra de parte de la tibia) es menor de 500 será el dato correspondiente al *heading*, ya que este se encuentra entre 0 y 360. Si tib1 está entre 815 y 1200 será el dato correspondiente al *pitch,* porque este está comprendido entre -180 y 180, y se le da un poco de margen. Y, por último, como al *roll* se le han sumado 2000, todo valor que esté por encima de 1999 corresponderá a este.

E. Calibración

El sensor BNO055 se tiene que calibrar para poder ser utilizado correctamente. Es decir, para que se apliquen los filtros que lleva interiormente, primero debe tener en cuenta que hay en su alrededor. Para esto hay que seguir unos pasos concretos.

- Para calibrar el giroscopio hay que mantener el dispositivo quieto durante unos segundos en cualquier posición.
- Calibrar el acelerómetro es un poco más largo, ya que hay que colocarlo en posiciones diferentes entre ellas con ángulos de 45 grados durante unos segundos en cada una de ellas.
- Por último, para calibrar el magnetómetro es suficiente con mover el dispositivo haciendo ochos por el aire.

Pero, el dispositivo tiene que saber cuando está calibrado y puede interpretar los datos como correctos y cuando tiene que ignorarlos.

Para ello, los dispositivos periféricos no envían datos al central si no están todas sus partes calibradas, es decir, el acelerómetro, giroscopio y magnetómetro.

Esto es posible llevarlo a cabo porque en las librerías de estos sensores existe la siguiente función, que devuelve números entre el 0 y el 3 dependiendo de si el dispositivo está completamente calibrado, siendo esto un 3, o nada calibrado, siendo esto un 0.

```
uint8_t system, gyro, accel, mag = 0;
bno.getCalibration(&system, &gyro, &accel, &mag);
```

Figura 59. Parte del código donde se comprueba la calibración de los sensores.

Por tanto, los dispositivos del fémur y la tibia no mandarán al dispositivo central los datos correspondientes a los ángulos si la función “bno.getcalibration()” no da un valor de 3.

Además, en el código central también se tiene que añadir otra parte, porque en el magnetómetro suele desaparecer la calibración, por tanto, hay que volver a saber si se están cogiendo los datos correctos. Al inicio del programa se ha introducido una variable global de tipo booleano llamada *calibrated*:

```
bool calibrated = false;
```

Figura 60. Introducción de la variable booleana “calibrated” en el código.

Y, más adelante, en el *void loop* se ha implementado el siguiente código:

```
while (calibrated == false){
    tft.fillRect(200, 60, 240, 180, HX8357_BLACK);
    tft.fillRect(2, 180, 240, 240, HX8357_BLACK);
    tft.setCursor(2, 180);
    tft.setTextColor(HX8357_WHITE);
    tft.setTextSize(2);
    tft.print("Por favor, pon los dispositivos juntos y toca la pantalla hasta que se pida otra cosa");

    delay(4000);
    TS_Point p = ts.getPoint();

   if (p.z > MINPRESSURE && p.z < MAXPRESSURE){

      Serial.print("OK");
      tft.fillRect(200, 60, 300, 300, HX8357_BLACK);
    tft.fillRect(2, 180, 300, 300, HX8357_BLACK);
    tft.setCursor(2, 180);
    tft.setTextColor(HX8357_WHITE);
    tft.setTextSize(2);
    tft.print( "Realiza los movimientos de calibración del magnetómetro y vuelve a poner los dispositivos juntos.");
    delay(100);

     delay(10000);
```

Figura 61. Parte del código en la que se pide hacer la calibración de los sensores.

Si el sistema detecta que se ha des calibrado el sensor avisa al usuario mediante un anuncio en la pantalla, y como la pantalla del dispositivo es táctil cuando el usuario esté listo dejará los sensores y tocará la pantalla para indicarle al sistema que está listo para calibrarlos. Entonces, cuando esté detecte que la pantalla está siendo tocada saltará el anuncio para que el usuario mueva los dispositivos un poco, ya que solo es necesario volver a calibrar el magnetómetro. Pero en el caso de que no haya sido suficiente volverá a saltar el anuncio y se volverá a llevar a cabo el mismo procedimiento.

F. Cálculo del ángulo relativo

Los datos que se quieren mostrar en la pantalla del dispositivo central, es decir, los datos que el cirujano necesita leer para poder realizar la operación correctamente son los ángulos relativos en cada uno de los tres planos anatómicos. Es decir, los datos finales necesarios por los que se han recogido los diferentes parámetros de los sensores son el *heading, pitch* y *roll* relativos de la tibia frente al fémur.

Cuando el dispositivo central ya ha recibido de cada uno de los sensores periféricos los datos de los ángulos y los ha clasificado tanto por de donde provienen como el tipo de ángulo que son, hay que llevar a cabo las operaciones necesarias para obtener los ángulos relativos, que son las siguientes:

En el plano coronal:

$$Heading = Heading\ (Tibia) - Heading\ (Femur)$$

En el plano sagital:

$$Pitch = Pitch\ (Tibia) - Pitch\ (Femur)$$

En el plano transversal:

$$Roll = Roll\ (Tibia) - Roll\ (Femur)$$

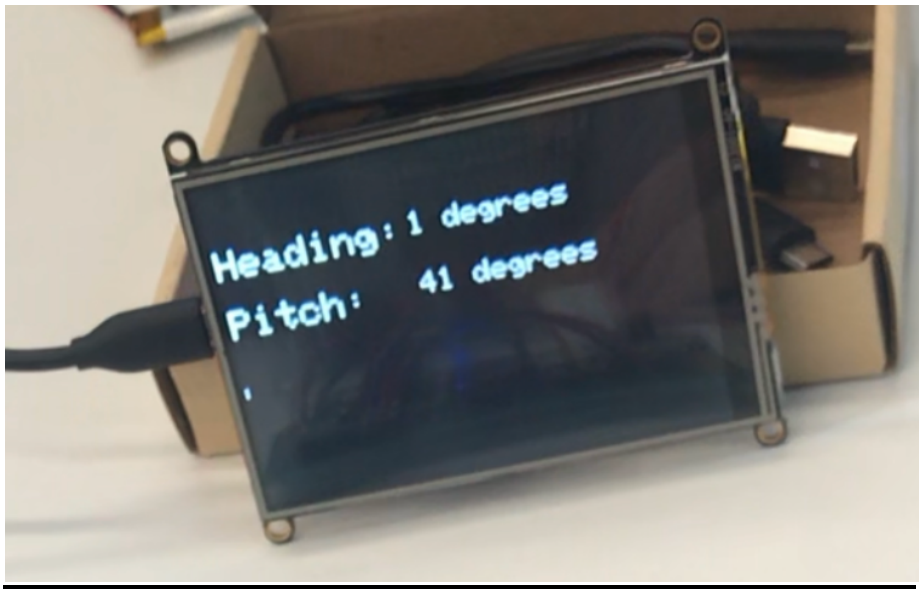


Figura 62. Muestra de la visión de la pantalla de los dos primeros datos.

CAPÍTULO 4. MANUAL DE FUNCIONAMIENTO

4.1. PUESTA EN MARCHA DEL DISPOSITIVO

En este apartado se va a explicar el funcionamiento para poner en marcha el dispositivo. Debido a que es simplemente un prototipo no se ha llegado a estudiar la manera de adherirlo a la pierna del paciente, no obstante, el cirujano para el que se desarrolló el proyecto expuso su intención de introducir los dos dispositivos periféricos dentro de dos bolsas diferentes de plástico y unirlos con cinta aislante al muñeco de simulación clínica con el que se debería probar, más adelante, su funcionamiento.

De esta manera, en este punto de desarrollo, para la puesta solo hace falta seguir dos pasos, la conexión de las baterías a los diferentes dispositivos y la calibración.

4.1.1. Conexión de baterías

Las placas Bluefruit nRF52 tienen un conector específico para conectar el tipo de baterías que se están utilizando, por lo que el procedimiento es tan sencillo como introducir el conector de la batería al conector de la placa.

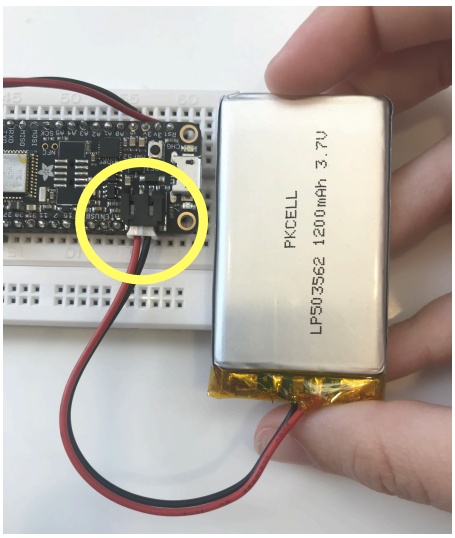


Figura 63. Conexión de la batería

4.1.2. Calibración

El primer paso a realizar es la calibración que, como se ha explicado en el apartado 1.3.2.2. E, se tiene que llevar a cabo en diferentes pasos.

Para calibrar el giroscopio basta con dejar el dispositivo estático durante unos segundos, como se muestra a continuación.

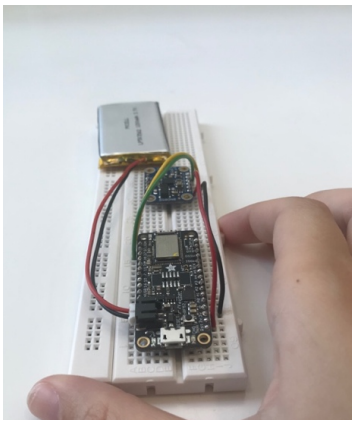


Figura 64. Calibración del giroscopio.

Posteriormente, se calibrará el acelerómetro, para lo que se tiene que ir dejando estático unos segundos cada dispositivo en ángulos de 45 grados como se muestra en las imágenes siguientes.

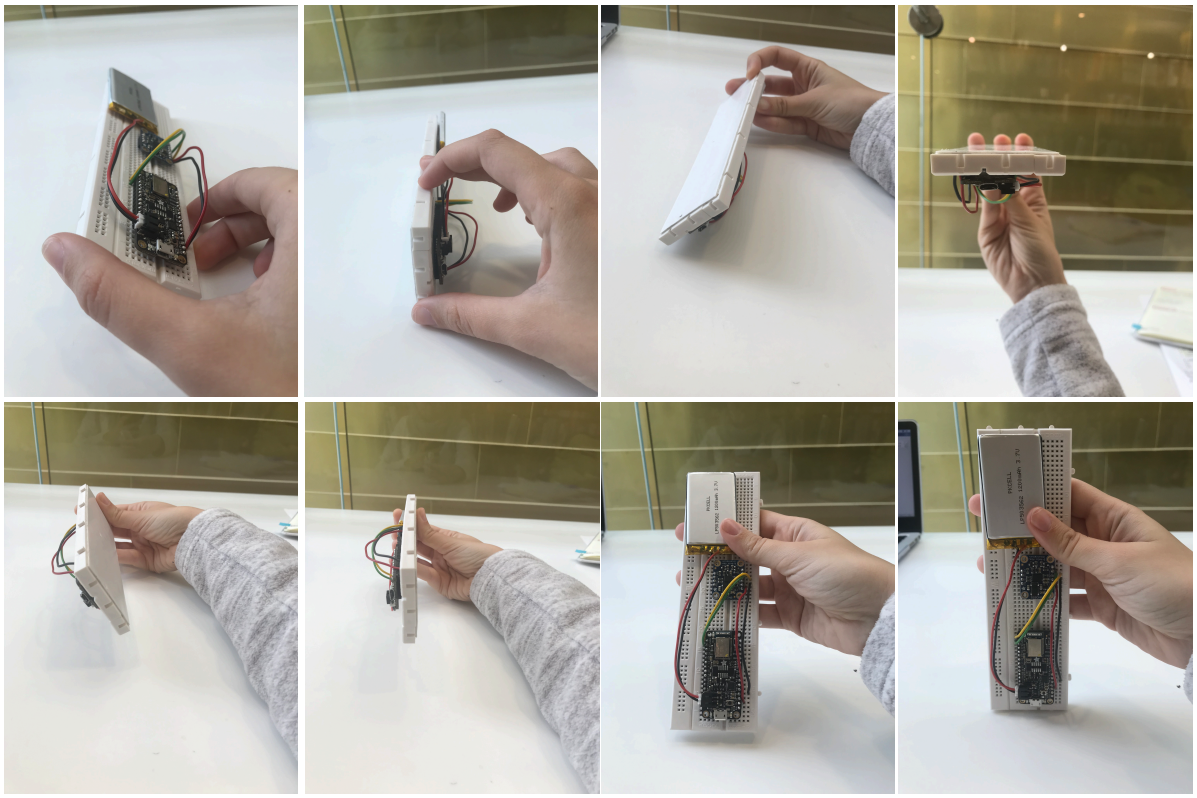


Figura 65. Calibración del acelerómetro

Y, por último, se calibra el magnetómetro, para lo que hay que mover los dispositivos haciendo ochos en el aire.

Figura 66. Calibración del magnetómetro

El magnetómetro es un sensor que necesita ser calibrado con frecuencia, debido a que reciben interferencias muy fácilmente con cualquier material metálico que haya alrededor. Por tanto, cuando el dispositivo se utiliza en una habitación en la que se ha utilizado unas horas antes sin haber sido apagado, el giroscopio y acelerómetro seguirán calibrados, pero habrá que volver a hacer los movimientos en forma de ochos en el aire que caracterizan la calibración del magnetómetro.

4.2. CARGA DE BATERÍA DEL DISPOSITIVO

Las placas Bluefruit nRF52 cuentan con una conexión de microUSB por la que se realiza la conexión al ordenador para cargar el código implementado en Arduino IDE. Este conector también puede utilizarse para cargar la batería conectada al dispositivo, por lo que es un procedimiento similar a la carga de un dispositivo móvil.

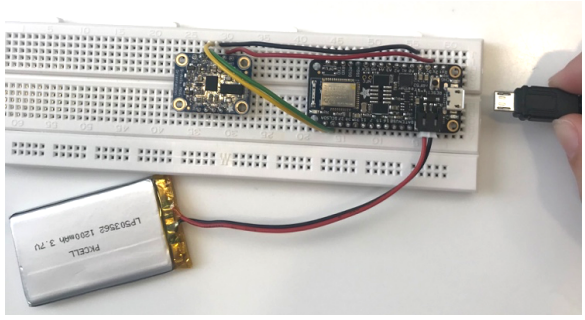


Figura 67. Conector microUSB

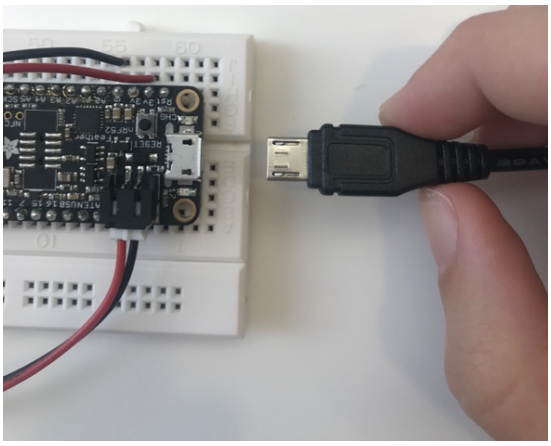


Figura 68. Conector microUSB

Adicionalmente, cuando la batería está cargándose, en la placa Bluefruit se enciende una luz naranja que indica la carga.

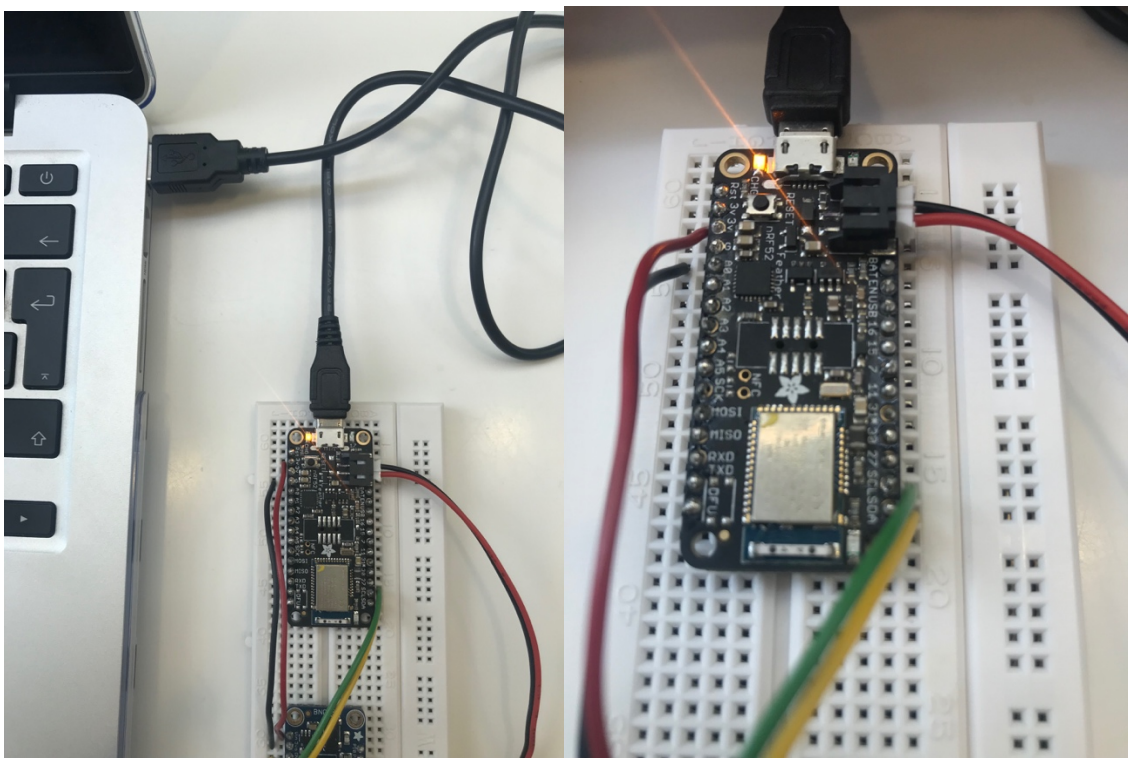


Figura 69. Conexión de la placa al ordenador para cargar. Luz naranja indicadora de carga.

CAPÍTULO 5. CONCLUSIONES Y LINEAS FUTURAS

Después del trabajo realizado para llevar a cabo este proyecto, se han alcanzado con éxito los objetivos que se planteaban al inicio del mismo. De esta manera, se han cumplido los siguientes puntos:

- Se ha diseñado un prototipo de dispositivo capaz de medir el ángulo entre dos puntos situados en los huesos fémur y tibia durante una artroplastia de rodilla. Para ello se ha utilizado un método de tracking híbrido compuesto por acelerómetros, giroscopios y magnetómetros, con los filtros necesarios para que el error sea mínimo.
- Aunque en este punto, el prototipo no es tan preciso como los sistemas de alineación por navegación, sí es más preciso que los sistemas basados en guías intramedulares y extramedulares, e igual de sencillo, siendo este un punto clave en la realización del diseño.
- Se ha desarrollado un sistema con conexiones Bluetooth y baterías incorporadas para minimizar los cables, consiguiendo de esta manera un dispositivo completamente inalámbrico en el momento de su uso.
- La interfaz de visualización es una pantalla TFT en la que se muestran los resultados sin tener que realizar ninguna complicada configuración.

Así, el desarrollo de este trabajo abarca desde la búsqueda bibliográfica, tanto de sistemas de tracking útiles para utilizar en cirugías de recambio de rodilla como de interfaces de conexión y otra electrónica que se ha utilizado, hasta la implementación de un prototipo con el método elegido.

A causa de las restricciones de tiempo y la falta de recursos, este prototipo no ha podido ser probado en ningún paciente o maniquí de simulación por el momento, no obstante, se ha comprobado el correcto funcionamiento situando los dispositivos periféricos (tibial y femoral) en barras metálicas simulando los huesos, y el resultado ha sido el deseado. Además, en los próximos meses se llevarán a cabo las pruebas necesarias en maniquíes de simulación.

Asimismo, como este dispositivo está pensado para utilizarse juntamente con el método de alineación con guías medulares, la parte femoral debe ir introducida dentro del hueso, por lo que, para que el dispositivo sea completamente apto para su utilización en cirugías reales, esta parte debe de tener un diámetro máximo de 12mm y longitud máxima de 200mm. Sin embargo, la miniaturización del dispositivo conlleva el diseño de placas PCB que debe ser realizado por un especialista electrónico.

Por otra parte, debido a la naturaleza del sensor magnetómetro, este prototipo es susceptible a interferencias por parte de materiales metálicos, lo que causa la pérdida de calibración del

dispositivo. De este punto se concluye que, una futura mejora a realizar en el proyecto es incluir en ambos dispositivos periféricos una memoria que no interfiera con el Bluetooth y las demás interfaces de comunicación que se están utilizando. Esta podría ser una memoria FRAM, y en ella se debería almacenar la calibración inicial de los sensores. Así, aunque imprevistamente se encuentren materiales metálicos cerca de los dispositivos, estos siempre utilizarán la calibración obtenida al inicio de la cirugía, por lo que los datos serán coherentes con la realidad.

En cuanto a la utilidad, aunque este dispositivo está diseñado para utilizarse en artroplastias de rodilla, debido a que su principal función es medir el ángulo relativo entre los huesos de la articulación, puede servir para otras aplicaciones en las que sea necesario conseguir una alineación determinada.

CAPÍTULO 6. BIBLIOGRAFÍA

Ada, L (2019). Adafruit 3,5" 480x320 TFT FeatherWing. Adafruit learning System.

American academy of orthopaedic surgeons (2019). Total knee replacement. Illinois: www. orthoinfo.aaos.org. https://orthoinfo.aaos.org/en/treatment/total-knee-replacement/

Angulo Pueyo E, Ridao Lopez M, Martínez Lizaga N, Seral Rodríguez M, Bernal-Delgado E (2014). Atlas de variaciones en la práctica médica: Ficha VPM Artroplastia de rodilla. Zaragoza (Variaciones en la pràctica médica (VPM).

Arduino (2019). Arduino Products. www.arduino.cc. https://www.arduino.cc/en/Main/Products

Burdea, G. C., & Coiffet, P. (2003). *Virtual reality technology*. John Wiley & Sons.

Bbraun sharing expertise. *OrthoPilot® Sistema de Navegación*. España: Bbraun.es. https://www.bbraun.es/es/products/b/orthopilot-sistemadenavegacion.html.

Cabezas Granado, L. M. (2010). Redes Inalámbricas. *Madrid:(Anaya multimedia)*.

Carletti, E. J. (2007). Comunicación–Bus I 2 C. *Robots Argentina.*

Castells, X., Comas, M., Guerrero, R., Espallargues, M., Allepuz, A., & Sabatés, S. (2014). Impacto de la cirugía para el recambio de prótesis de rodilla en el Sistema Nacional de Salud. *Agència de Qualitat i Avaluació Sanitàries de Catalunya*.

Clemens, U., & Miehlke, R. K. (2003). Experience using the latest OrthoPilot TKA software: a comparative study. *Surgical technology international*, *11*, 265-273.

Doña, D. Introducción a los sistemas informáticos. España: www.danieldona.com. https://www.danieldona.com/informatica%20basica/2%20sistemas%20informaticos.pdf

Drake, R., Vogl, W. and Mitchell, A. Gray's anatomy for students. 2nd ed. pp.575, 576.

Ecured (2018). LCD (Pantalla de cristal líquido). España: www.edured.cu. https://www.ecured.cu/LCD_(pantalla_de_cristal_l%C3%ADquido)

ElProCus. (2012). Different Types of Wireless Communication with Applications. India: www.elprocus.com. https://www.elprocus.com/types-of-wireless-communication-applications/#comments

Faragher, R., 2012. Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation. IEEE Signal Processing Magazie, 01 September, pp. 128-132.

García-Porrero, J. A., & Hurlé, J. M. (2013). Anatomía humana.

Gómez Vallejo, J. (2011). El recambio en la artroplástia total de rodilla: Aspectos clínicos, radiológicos, factores de supervivencia y calidad de vida.

Hart, R., Janeček, M., Chaker, A., & Buček, P. (2003). Total knee arthroplasty implanted with and without kinematic navigation. *International orthopaedics*, *27*(6), 366-369.

Hinarejos Gómez, Pedro. Técnica quirúrgica para cirugía protésica de rodilla.

Informática moderna (2019). La pantalla TFT. España: www.informaticamoderna.com. http://www.informaticamoderna.com/Pantalla_TFT.htm

Instruments, T (2015). TI designs. UART to Bluetooth low energy (BLE) Bridge Design Guida.

Lavernia, C., & Alcerro, J. (2008). Artroplastia total de rodilla. Act Pos Méd Gen, 13(7), 6-11.

Mary, R. (2010). Wireless Communication and types. www.engineersgarage.com. https://www.engineersgarage.com/articles/wireless_communication

Michael, M. S. (1992). U.S. Patent No. 5,140,679. Washington, DC: U.S. Patent and Trademark Office.

Nordic semiconductor. (2016). nRF52832 - Product Specification v1.0. 1st ed.

Osio, J. R., Antonini, L., Aróztegui, W., & Rapallini, J. A. (2011). Descripción General de un Microcontrolador (Módulos de Comunicación). Universidad Nacional de la Plata

Patkar, M. (2018). What is Bluetooth? 10 common questions, asked and answered. https://www.makeuseof.com/tag/what-is-bluetooth/

PROTESIS TOTAL DE RODILLA -Instrumental y técnica- Dr.Gonzalo Mora. (2012). [video] Logroño, La Rioja, España: DrGonzalo Mora.

Ramírez, Á. S. (2003). El filtro de kalman. *Documento de trabajo del Banco Central de Costa Rica, elaborado en la División Económica, Departamento de Investigaciones Económicas.*

Raspberry Pi (2019). Teach, Learn, and Make with Raspberry Pi. https://www.raspberrypi.org

Ray, B. (2015). 5 types of wireless Technology for the IoT. Oxford: www.link-labs.com . https://www.link-labs.com/blog/types-of-wireless-technology

Rodríguez, J. L. M., Ochoa, D. R. H., Henríquez, J. A. V., & Méndez (2012), C. A. C. COMEC 2012.

Scrosati, B., Abraham, K. M., van Schalkwijk, W. A., & Hassoun, J. (Eds.). (2013). *Lithium batteries: advanced technologies and applications* (Vol. 58). John Wiley & Sons.

Sensortec, B. (2016). Intelligent 9-axis absolute orientation sensor. *BNO055 datasheet.*

Sensor City (2019). About Sensor City. Liverpool: www.sensorcity.com. https://www.sensorcity.co.uk/about/

Townsend, K. (2019). Bluefruit nRF52 Feather Learning Guide. Adafruit learning System.

Welch, G., & Bishop, G. (1995). An introduction to the Kalman filter.

PRESUPUESTOS

1. INTRODUCCIÓN

En este documento del trabajo se presenta el presupuesto del proyecto realizado. En él se va a analizar el coste que conlleva realizar el proyecto de diseño y desarrollo de un sistema de alineación para artroplastias de rodilla.

Con objetivo de obtener el presupuesto final para la realización de un dispositivo de este tipo se va a realizar un presupuesto de mano de obra, materiales y equipos y se van a obtener precios parciales, unitarios y descompuestos para la obtención de un presupuesto final de ejecución por contrata.

Para la realización del proyecto se ha considerado que la mano de obra es un ingeniero biomédico recién titulado, con un sueldo de 15€ por hora.

En cuanto al tiempo de vida útil de los equipos empleados, se ha considerado ocho años para el ordenador MacBook Pro, tres años para la impresora 3D y 10 años para la estación de soldadura.

Asimismo, se tiene en cuenta que, habiendo en el año 365 días, de los cuales 115 son festivos y fines de semana, y que el empleado que utiliza estos dispositivos tiene 30 días de vacaciones anuales, quedan 224 días de trabajo al año en los que se trabajan 8 horas diarias. De esta manera se obtiene que las horas de amortización de los equipos anteriores, por año, son 1800.

Por otra parte, los softwares Microsoft Office y Fusion 360 tienen una duración determinada de tiempo instalado, por lo que se ha calculado que su amortización anual son las 24 horas del día de los 365 días.

Finalmente, para el presupuesto de ejecución por contrata se ha añadido un 13% al presupuesto de ejecución material como gastos generales de la empresa, así como un 6% más de beneficio industrial y un 21% de IVA, ya que no se ha incluido anteriormente.

2. PRESUPUESTO DE MANO DE OBRA

Nº	Código	Denominación de la mano de obra	Precio	Horas	Total
1	MO.IB	Ingeniero biomédico recien titulado	15 €/h	513,60 €	7.704,00 €
				Total mano de obra:	**7.704,00 €**

3. PRESUPUESTO DE MATERIALES Y EQUIPOS

Nº	Código	Denominación del material/equipo/software	Precio	Cantidad	Factor de amortización	Total
1	MAT.ARD	Placa Arduino Bluefruit feather nRF52832	18,88 €	3 u	-	56,64 €
2	MAT.SEN	Sensor Adafruit BNO055	26,67 €	2 u	-	53,34 €
3	MAT.PAN	Pantalla Adafruit TFT Feather Wing 3.5 Touchscreen	30,22 €	1 u	-	30,22 €
4	MAT.BB	Placa de prototipo BreadBoard	4,00 €	2 u	-	8,00 €
5	MAT.CAB	Cable	0,02 €	8 u	-	0,16 €
6	MAT.BAT	Batería de litio de 3,7V - 1200 mAh	10,77 €	3 u	-	32,31 €
7	MAT.MICRO	Cable microUSB	3,20 €	3 u	-	9,60 €
8	MAT.PLA	PLA para impresión	16,99 € / kg	133 g	-	2,26 €
9	EQ.ORD	Ordenador MacBook Pro	1.200,00 €	499,5 h	0,0350	41,63 €
10	SOF.ARD	Software Arduino IDE	0 €	200 h	-	0 €
11	EQ.IMP	Impresora Ultimaker 3	3.666,30 €	22 h	0,0040	14,94 €
12	SOF.FUS	Software Fusion 360 (Autodesk)	502,15 €	20 h	0,0028	1,40 €
13	SOF.OFF	Paquete Office	123,14 €	160 h	0,0046	0,56 €
14	EQ.SOLD	Estación de soldadura	65,00 €	2 h	0,0001	0,0072 €
15	SOF.CUR	Software Ultimaker Cura	0 €	0,5 h	-	0 €
		Total materiales, equipos y software:				**251,06 €**

4. PRESUPUESTOS PARCIALES

4.1. Planificación del proyecto y conocimiento del estado del arte

Nº	Unidad	Denominación	Cantidad	Precio	Total
1.1	h	Reuniones con la empresa Knee innovations para planificar el proyecto	2 h	15,000 €	30,000 €
1.2	h	Revisión de la literatura sobre artroplastias de rodilla	35 h	15,083 €	527,905 €
1.3	h	Revisión de la literatura sobre métodos de alineación o *tracking*	6 h	15,000 €	90,000 €
1.4	u	Selección y adquisición de los sensores y las placas necesarias para la realización del proyecto	1 u	310,777 €	310,777 €
1.5	u	Instalación de los programas necesarios para la realización del proyecto y aprendizaje del lenguaje de programación	1 u	1.131,250 €	1.131,250 €
		Total presupuesto parcial nº1 - Planificación del proyecto y conocimiento del estado del arte:			**2.089,93 €**

4.2. Montaje de la parte electrónica

Nº	Unidad	Denominación	Cantidad	Precio	Total
2.1	u	Selección de los cables necesarios	1 u	1,660 €	1,660 €
2.2	h	Soldadura de las diferentes placas de Arduino y sensores	2 h	15,004 €	30,007 €
		Total presupuesto parcial nº2 - Montaje de la parte electrónica:			**31,67 €**

4.3. Desarrollo del código de los microcontroladores

Nº	Unidad	Denominación	Cantidad	Precio	Total
3.1	h	Descarga e instalación de las bibliotecas necesarias para la realización de los códigos	3 h	15,083 €	45,240 €
3.2	h	Realización de los códigos de los diferentes dispositivos con el software Arduino IDE	190 h	15,083 €	2.865,770 €
3.3	h	Comprobación y corrección de errores para el funcionamiento de los códigos	8 h	15,083 €	120,664 €
		Total presupuesto parcial nº3 - Desarrollo del código de los microcontroladores:			**3.031,67 €**

4.4. Diseño e impresión de la envoltura física del dispositivo

Nº	Unidad	Denominación	Cantidad	Precio	Total
4.1	h	Diseño de la envoltura física de cada una de las partes del dispositivo con el software Fusion 360	20 h	15,140 €	302,800 €
4.2	h	Preparación de los diseños para ser impresos	0,5 h	15,083 €	7,542 €
4.3	u	Impresión 3D de la envoltura física de los dispositivos	1 u	17,190 €	17,190 €
4.4	h	Limpieza de las piezas impresas y montaje de los dispositivos en ellas	4 h	15,000 €	60,000 €
		Total presupuesto parcial nº4 - Diseño e impresión de la envoltura física del dispositivo:			**387,53 €**

4.5. Desarrollo de los documentos del proyecto

Nº	Unidad	Denominación	Cantidad	Precio	Total
5.1	h	Redacción de los documentos del proyecto	150 h	15,086 €	2.262,900 €
5.2	h	Revisión de los documentos y corrección de errores	10 h	15,086 €	150,860 €
		Total presupuesto parcial nº5 - Desarrollo de los documentos del proyecto:			**2.413,76 €**

5. PRECIOS UNITARIOS

Nº	Denominación	Importe	
		En cifra (€)	En letra (€)
	1 PLANIFICACIÓN DEL PROYECTO Y CONOCIMIENTO DEL ESTADO DEL ARTE		
1.1	h Reuniones con la empresa Knee innovations para planificar el proyecto	15,00 €	Quince euros
1.2	h Revisión de la literatura sobre artroplastias de rodilla	15,08 €	Quince euros con ocho céntimos
1.3	h Revisión de la literatura sobre métodos de alineación o *tracking*	15,00 €	Quince euros
1.4	u Selección y adquisición de los sensores y las placas necesarias para la realización del proyecto	310,78 €	Tres cientos diez euros con setenta y ocho céntimos
1.5	u Instalación de los programas necesarios para la realización del proyecto y aprendizaje del lenguaje de programación	1.131,25 €	Mil ciento trenta y un euros con veinticinco céntimos
	2 MONTAJE DE LA PARTE ELECTRÓNICA		
2.1	u Selección de los cables necesarios	1,66 €	Un euro con sesenta y seis céntimos
2.2	h Soldadura de las diferentes placas de Arduino y sensores	15,00 €	Quince euros
	3 DESARROLLO DEL CÓDIGO DE LOS MICROCONTROLADORES		
3.1	h Descarga e instalación de las bibliotecas necesarias para la realización de los códigos	15,08 €	Quince euros con ocho céntimos
3.2	h Realización de los códigos de los diferentes dispositivos con el software Arduino IDE	15,08 €	Quince euros con ocho céntimos
3.3	h Comprobación y corrección de errores para el funcionamiento de los códigos	15,08 €	Quince euros con ocho céntimos
	4 DISEÑO E IMPRESIÓN DE LA ENVOLTURA FÍSICA DEL DISPOSITIVO		
4.1	h Diseño de la envoltura física de cada una de las partes del dispositivo con el software Fusion 360	15,14 €	Quince euros con catorce céntimos
4.2	h Preparación de los diseños para ser impresos	15,08 €	Quince euros con ocho céntimos
4.3	Impresión 3D de la envoltura física de los dispositivos	17,19 €	Diecisiete euros con diecinueve céntimos
4.4	h Limpieza de las piezas impresas y montaje de los dispositivos en ellas	15,00 €	Quince euros
	5 DESARROLLO DE LOS DOCUMENTOS DEL PROYECTO		
5.1	h Redacción de los documentos del proyecto	15,09 €	Quince euros con nueve céntimos
5.2	h Revisión de los documentos y corrección de errores	15,09 €	Quince euros con nueve céntimos

6. PRECIOS DESCOMPUESTOS

Nº	Ud	Denominación					Total
		1 PLANIFICACIÓN DEL PROYECTO Y CONOCIMIENTO DEL ESTADO DEL ARTE					
1.1	h	**Reuniones con la empresa Knee innovations para planificar el proyecto**					
		MO.IB	1h		Ingeniero biomédico	15 €/h	15,000 €
						Precio total por h:	**15,000 €**
1.2	h	**Revisión de la literatura sobre artroplastias de rodilla**					
		MO.IB	1 h		Ingeniero biomédico	15 €/h	15,000 €
		EQ.ORD	1 h	0,00007	Ordenador MacBook Pro	1.200,00 €	0,083 €
						Precio total por h:	**15,083 €**
1.3	h	**Revisión de la literatura sobre métodos de alineación o tracking**					
		MO.IB	1 h		Ingeniero biomédico	15 €/h	15,000 €
						Precio total por h:	**15,000 €**
1.4	u	**Selección y adquisición de los sensores y las placas necesarias para la realización del proyecto**					
		MO.IB	8 h		Ingeniero biomédico	15 €/h	120,000 €
		EQ.ORD	8 h	0,00056	Ordenador MacBook Pro	1.200,00 €	0,667 €
		MAT.ARD	3 u		Placa Arduino Bluefruit feather nRF52832	18,88 €	56,640 €
		MAT.SEN	2 u		Sensor Adafruit BNO055	26,67 €	53,340 €
		MAT.PAN	1 u		Pantalla Adafruit TFT Feather Wing 3.5 Touchscreen	30,22 €	30,220 €
		MAT.BB	2 u		Placa de prototipo BreadBoard	4,00 €	8,000 €
		MAT.BAT	3 u		Batería de litio de 3,7V - 1200 mAh	10,77 €	32,310 €
		MAT.MICRO	3 u		Cable microUSB	3,20 €	9,600 €
						Precio total por u:	**310,777 €**

1.5 u **Instalación de los programas necesarios para la realización del proyecto y aprendizaje del lenguaje de programación**

MO.IB	75 h		Ingeniero biomédico	15 €/h	1.125,000 €
EQ.ORD	75 h	0,0052	Ordenador MacBook Pro	1.200,00 €	6,250 €
SOF.ARD	1 u	-	Software Arduino IDE	0 €	0,000 €
				Precio total por u:	**1.131,250 €**

2 MONTAJE DE LA PARTE ELECTRÓNICA

2.1 u **Selección de los cables necesarios**

MO.IB	0,1 h	Ingeniero biomédico	15 €/h	1,500 €
MAT.CAB	8 u	Cable	0,02 €	0,160 €
			Precio total por u:	**1,660 €**

2.2 h **Soldadura de las diferentes placas de Arduino y sensores**

MO.IB	1 h		Ingeniero biomédico	15 €/h	15,000 €
EQ.SOLD	1 h	0,0006	Estación de soldadura	65 €	0,004 €
				Precio total por h:	**15,004 €**

3 DESARROLLO DEL CÓDIGO DE LOS MICROCONTROLADORES

3.1 h **Descarga e instalación de las bibliotecas necesarias para la realización de los códigos**

MO.IB	1 h		Ingeniero biomédico	15 €/h	15,000 €
EQ.ORD	1 h	0,00007	Ordenador MacBook Pro	1.200,00 €	0,083 €
SOF.ARD	1 h	-	Software Arduino IDE	0 €	0,000 €
				Precio total por h:	**15,083 €**

3.2 h **Realización de los códigos de los diferentes dispositivos con el software Arduino IDE**

MO.IB	1 h		Ingeniero biomédico	15 €/h	15,000 €
EQ.ORD	1 h	0,00007	Ordenador MacBook Pro	1.200,00 €	0,083 €
SOF.ARD	1 h	-	Software Arduino IDE	0 €	0,000 €
				Precio total por h:	**15,083 €**

3.3 h **Comprobación y corrección de errores para el funcionamiento de los códigos**

MO.IB		1 h	Ingeniero biomédico	15 €/h	15,000 €
EQ.ORD	1 h	0,00007	Ordenador MacBook Pro	1.200,00 €	0,083 €
SOF.ARD	1 h	-	Software Arduino IDE	0 €	0,000 €
				Precio total por h:	**15,083 €**

4 DISEÑO E IMPRESIÓN DE LA ENVOLTURA FÍSICA DEL DISPOSITIVO

4.1 h **Diseño de la envoltura física de cada una de las partes del dispositivo con el software Fusion 360**

MO.IB		1 h	Ingeniero biomédico	15 €/h	15,000 €
EQ.ORD	1 h	0,00007	Ordenador MacBook Pro	1.200,00 €	0,083 €
SOF.FUS	1 h	0,00011	Software Fusion 360 (Autodesk)	502,15 €	0,057 €
				Precio total por h:	**15,140 €**

4.2 h **Preparación de los diseños para ser impresos**

MO.IB		1 h	Ingeniero biomédico	15 €/h	15,000 €
EQ.ORD	1 h	0,00007	Ordenador MacBook Pro	1.200,00 €	0,083 €
SOF.CUR	1 h	-	Software Ultimaker Cura	0 €	0,000 €
				Precio total por h:	**15,083 €**

4.3 u **Impresión 3D de la envoltura física de los dispositivos**

EQ.IMP	22 h	0,0041	Impresora Ultimaker 3	3.666,30 €	14,930 €
MAT.PLA		133 g	PLA para impresión	16,99 €/kg	2,260 €
				Precio total por u:	**17,190 €**

4.4 h **Limpieza de las piezas impresas y montaje de los dispositivos en ellas**

MO.IB		1 h	Ingeniero biomédico	15 €/h	15,000 €
				Precio total por h:	**15,000 €**

5 DESARROLLO DE LOS DOCUMENTOS DEL PROYECTO

5.1 h **Redacción de los documentos del proyecto**

MO.IB		1 h	Ingeniero biomédico	15 €/h	15,000 €
EQ.ORD	1 h	0,00007	Ordenador MacBook Pro	1.200,00 €	0,083 €
SOF.OFF	1 h	0,000028	Paquete Office	123,14 €	0,004 €
				Precio total por h:	**15,087 €**

5.2 h **Revisión de los documentos y corrección de errores**

MO.IB		1 h	Ingeniero biomédico	15 €/h	15,000 €
EQ.ORD	1 h	0,00007	Ordenador MacBook Pro	1.200,00 €	0,083 €
SOF.OFF	1 h	0,000028	Paquete Office	123,14 €	0,004 €
				Precio total por h:	**15,087 €**

7. PRESUPUESTO DE EJECUCIÓN POR CONTRATA

Capítulo	**Importe (€)**
Capítulo 1. Planificación del proyecto y conocimiento del estado del arte	2.089,93 €
Capítulo 2. Montaje de la parte electrónica	31,67 €
Capítulo 3. Desarrollo del código de los microcontroladores	3.031,67 €
Capítulo 4. Diseño e impresión de la envoltura física del dispositivo	387,53 €
Capítulo 5. Desarrollo de los documentos del proyecto	2.413,76 €
Presupuesto de ejecución material	7.954,56 €
Gastos generales (13%)	1.034,09 €
Beneficio industrial (6%)	477,27 €
Suma	9.465,93 €
IVA (21%)	1.987,84 €
Presupuesto de ejecución por contrata	**11.453,77 €**

ANEXOS

ANEXO 1. CÓDIGOS DE ARDUINO IDE PARA LOS TRES DISPOSITIVOS

1. INTRODUCCIÓN

En este anexo se van a incluir los códigos implementados en los tres dispositivos que configuran el dispositivo final desarrollado para el presente proyecto.

Estos códigos fueron creados a partir de unas librerías descargadas en octubre de 2018, que contienen funciones internas de cada placa. Estas librerías se van actualizando continuamente para realizar cambios y mejoras, por lo que es posible que el código no se pueda utilizar de manera exacta con librerías descargadas en tiempos futuros. No obstante, los cambios que afectarían a este código no son demasiados, por tanto, leyendo los ejemplos de las nuevas librerías se podrían encontrar fácilmente.

2. CÓDIGO DEL DISPOSITIVO CENTRAL

```
#include <Adafruit_GFX.h>
#include "Adafruit_HX8357.h"
#include <bluefruit.h>
#include "TouchScreen.h"
#include <SPI.h>
#include <Adafruit_STMPE610.h>


// Definición de una estructura que contiene información de los dispositivos periférico
typedef struct
{
  char name[32];

  uint16_t conn_handle;

  BLEClientUart bleuart;
} prph_info_t;


// Initialización de las diferentes variables
String tibia[3];
String femur[3];
float htibia;
float ptibia;
float rtibia;
float hfemur;
float pfemur;
float rfemur;
float ftibia;
float ffemur;
int resth;
int restp;
float calibh;
float calibp;
bool calibrated = false;



// Datos de calibración para la pantalla táctil
#define TS_MINX 3800
#define TS_MINY 100
#define TS_MAXX 100
#define TS_MAXY 3750

#define MINPRESSURE 10
#define MAXPRESSURE 1000

// Definición de los pins de la pantalla
#ifdef ARDUINO_NRF52_FEATHER
#define TFT_DC   11
#define TFT_CS   31
#define STMPE_CS 30
#define SD_CS    27
#endif

#define TFT_RST -1

Adafruit_HX8357 tft = Adafruit_HX8357(TFT_CS, TFT_DC, TFT_RST);


Adafruit_STMPE610 ts = Adafruit_STMPE610(STMPE_CS);
```

```
prph_info_t prphs[BLE_CENTRAL_MAX_CONN];

// Definición del control de tiempo de parpadeo del LED rojo
SoftwareTimer blinkTimer;
uint8_t connection_num = 0;

// Función de configuración
void setup()
{
   // Inicio del serial por si el dispositivo estuviera conectado al ordenador y fuera necesario
  Serial.begin(115200);

    if (!ts.begin()) {
    Serial.println("Couldn't start touchscreen controller");
    while (1);
  }
  Serial.println("Touchscreen started");

 // Se inicializa la pantalla, se hace el fondo de la pantalla negra y las letras blancas.
 // Fija los enunciados de los datos ("Heading:", "Pitch:" y "Roll:") en la pantalla.
  tft.begin(HX8357D);
  yield();
  tft.setRotation(1);
  tft.fillScreen(HX8357_BLACK);
  tft.setCursor(2, 60);
  tft.setTextSize(4);
  tft.setTextColor(HX8357_WHITE);
  tft.println("Heading: ");
  tft.setCursor(2, 120);
  tft.println("Pitch: ");
  tft.setCursor(2, 180);
  tft.println("Roll: ");


  // Inicialización del parpadeo del LED rojo
  blinkTimer.begin(100, blink_timer_callback);
  blinkTimer.start();


  // Inicialización de la placa Bluefruit (Arduino)
  Bluefruit.begin(0, 4);

  // Se le da un nombre al dispositivo
  Bluefruit.setName("Central device");

  // Inicio de la búsqueda de dispositivos periféricos
  for (uint8_t idx = 0; idx < BLE_CENTRAL_MAX_CONN; idx++)
  {

    prphs[idx].conn_handle = BLE_CONN_HANDLE_INVALID;

    // Se inicia del BLE UART y se hacen dos llamadas para conectar los dos dispositivos periféricos que se espera encontrar
    prphs[idx].bleuart.begin();
    prphs[idx].bleuart.setRxCallback(bleuart_rx_callback_one);
    prphs[idx].bleuart.setRxCallback(bleuart_rx_callback_two);

  }

  // Callbacks for Central
  Bluefruit.Central.setConnectCallback(connect_callback);
  Bluefruit.Central.setDisconnectCallback(disconnect_callback);
```

```
  Bluefruit.Scanner.setRxCallback(scan_callback);
  Bluefruit.Scanner.restartOnDisconnect(true);
  Bluefruit.Scanner.setInterval(160, 80);       // in units of 0.625 ms
  Bluefruit.Scanner.useActiveScan(false);       // Don't request scan response data
  Bluefruit.Scanner.start(0);                   // 0 = Don't stop scanning after n seconds


}

// Función para escanear dispositivos
void scan_callback(ble_gap_evt_adv_report_t* report)
{
  // Check if advertising data contains the BleUart service UUID
  if ( Bluefruit.Scanner.checkReportForUuid(report, BLEUART_UUID_SERVICE) )
  {
    Serial.print("BLE UART service detected. Connecting ... ");

    // Connect to the device with bleuart service in advertising packet
    Bluefruit.Central.connect(report);
  }
}

// Función para conectar los dispositivos encontrados
void connect_callback(uint16_t conn_handle)
{

  int id  = findConnHandle(BLE_CONN_HANDLE_INVALID);

  if ( id < 0 ) return;

  prph_info_t* peer = &prphs[id];
  peer->conn_handle = conn_handle;

   // Obtención del nombre del dispositivo al que se conecta
  Bluefruit.Gap.getPeerName(conn_handle, peer->name, 32);

   // Si el dispositivo está conectado a un ordenador en el serial se muestra a quie dispositivo está conectado
  // Además sigue buscando otros dispositivos periféricos
  // Si no se encuentra nada lo informa
  Serial.print("Connected to ");
  Serial.println(peer->name);


  if ( peer->bleuart.discover(conn_handle) )
  {
    Serial.println("Dispositivo encontrado");
    peer->bleuart.enableTXD();

    Serial.println("Buscando más dispositivos periféricos...");
    Bluefruit.Scanner.start(0);

  } else
  {
    Serial.println("No se ha encontrado nada!");

    // Se desconecta porque no encuentra nada
    Bluefruit.Central.disconnect(conn_handle);
  }

  connection_num++;
}
```

```
// Función para cuando se desconecta un dispositivo. Muestra qué dispositivo y la razón
void disconnect_callback(uint16_t conn_handle, uint8_t reason)
{
  (void) conn_handle;
  (void) reason;

  connection_num--; // Se disminuyen el número de conexiones


  int id  = findConnHandle(conn_handle);
  if ( id < 0 ) return;


  prphs[id].conn_handle = BLE_CONN_HANDLE_INVALID;
  Serial.print(prphs[id].name);
  Serial.println(" disconnected!");
}

// Función para cuando se reciben datos mediante BLE UART de uno de los dispositivos
void bleuart_rx_callback_one(BLEClientUart& uart_svc)
{

  uint16_t conn_handle = uart_svc.connHandle();

  int id = findConnHandle(conn_handle);
  prph_info_t* peer = &prphs[id];

// Mientras el puerto UART esté disponible, si hay información en dicho puerto se recoge.
  while ( uart_svc.available() )
  {
    char buf[20 + 1] = { 0 };
   if ( uart_svc.read(buf, sizeof(buf) - 1) )
   {
      // Si la información viene de la tibia, se recogen los datos en un String
     if (strcmp(peer->name, "Tibia sensor") == 0) {
       for (int i = 0; i < 3; i++) {
         tibia[i] = String (buf);
       }

       // Se separa cada uno de los números en un punto flotante
       float tib1 = tibia[0].toFloat();
       float tib2 = tibia[1].toFloat();
       float tib3 = tibia[2].toFloat();

        // Dependiendo de su valor (porque en los códigos de los dispositivos periféricos se había sumado diferentes valores a las diferentes variables
       // Se distingue entre heading, pitch y roll
       if (tib1 < 500) {
         htibia = tib1;
       }
       if (tib1 > 815 && tib1 < 1200) {
         ptibia = tib1;
       }
        if ( tib1 > 1999){
             rtibia = tib1;
            }

       if (tib2 < 500) {
         htibia = tib2;
       }
       if (tib2 > 815 && tib2 < 1200) {
         ptibia = tib2;
```

```
      }
      if ( tib2 > 1999){
            rtibia = tib2;
           }


      if (tib3 < 500) {
        htibia = tib3;
      }
      if (tib3 > 815 && tib3 < 1200) {
        ptibia = tib3;
      }
       if ( tib3 > 1999){
            rtibia = tib3;
           }

    }

   // Lo mismo que anteriormente pero cuando los datos vienen del dispositivo del fémur
    if (strcmp(peer->name, "Femur sensor") == 0) {
      for (int j = 0; j < 3; j++) {
        femur[j] = String (buf);

      }
      float fem1 = femur[0].toFloat();
      float fem2 = femur[1].toFloat();
      float fem3 = femur[2].toFloat();

      if (fem1 < 500) {
        hfemur = fem1;
      }
       if (fem1 > 815 && fem1 < 1200) {
         pfemur = fem1;
       }

       if ( fem1 > 1999){
         rfemur = fem1;
       }

       if (fem2 < 500) {
         hfemur = fem2;
       }

       if (fem2 > 815 && fem2 < 1200) {
         pfemur = fem2;
       }

       if ( fem2 > 1999){
         rfemur = fem2;
            }

       if (fem3 < 500) {
         hfemur = fem3;
       }

       if (fem3 > 815 && fem3 < 1200) {
         pfemur = fem3;
       }

      if ( fem3 > 1999){
         rfemur = fem3;
            }
```

```
      }

    }

  }
}


// Función para recibir los datos del otro dispositivo periférico.
// Se realizan los mismos pasos que anteriormente
// En el caso de que anteriormente los datos provinieran del fémur ahora vendrán de la tibia y viceversa
void bleuart_rx_callback_two(BLEClientUart& uart_svc)
{
  uint16_t conn_handle = uart_svc.connHandle();

  int id = findConnHandle(conn_handle);
  prph_info_t* peer = &prphs[id];


  while ( uart_svc.available() )
  {
    char buf[20 + 1] = { 0 };
    if ( uart_svc.read(buf, sizeof(buf) - 1) )
    {


      if (strcmp(peer->name, "Tibia sensor") == 0) {
        for (int i = 0; i < 3; i++) {
          tibia[i] = String (buf);
       }

       float tib1 = tibia[0].toFloat();
       float tib2 = tibia[1].toFloat();
       float tib3 = tibia[2].toFloat();

       if (tib1 < 500) {
         htibia = tib1;
       }
       if (tib1 > 815 && tib1 < 1200) {
         ptibia = tib1;
       }
             if ( tib1 > 1999){
             rtibia = tib1;
            }

       if (tib2 < 500) {
         htibia = tib2;
       }
       if (tib2 > 815 && tib2 < 1200) {
         ptibia = tib2;
       }

        if ( tib2 > 1999){
             rtibia = tib2;
            }
       if (tib3 < 500) {
         htibia = tib3;
       }
       if (tib3 > 815 && tib3 < 1200) {
         ptibia = tib3;
```

```
        }

    if ( tib3 > 1999){
              rtibia = tib3;
             }
      }


      if (strcmp(peer->name, "Femur sensor") == 0) {
        for (int j = 0; j < 3; j++) {
          femur[j] = String (buf);
          // ffemur= femur[j].toFloat();
        }
        float fem1 = femur[0].toFloat();
        float fem2 = femur[1].toFloat();
        float fem3 = femur[2].toFloat();

        if (fem1 < 500) {
          hfemur = fem1;
        }
        if (fem1 > 815 && fem1 < 1200) {
          pfemur = fem1;
        }

         if ( fem1 > 1999){
              rfemur = fem1;
             }
        if (fem2 < 500) {
          hfemur = fem2;
        }
        if (fem2 > 815 && fem2 < 1200) {
          pfemur = fem2;
        }

         if ( fem2 > 1999){
              rfemur = fem2;
             }
        if (fem3 < 500) {
          hfemur = fem3;
        }
        if (fem3 > 815 && fem3 < 1200) {
          pfemur = fem3;
        }

     if ( fem3 > 1999){
              rfemur = fem3;
             }
      }

    }

  }

}

// Función principal de repetición
void loop(void)
{
  // Se comprueba si el dispositivo está conectado a algún periférico
  if ( Bluefruit.Central.connected() )
  {
    char buf[20 + 1] = { 0 };
```

```
// Si la variable booleana sigue en "false" en la pantalla saltan mensajes que piden calibrar los dispositivos

while (calibrated == false){
    tft.fillRect(200, 60, 240, 180, HX8357_BLACK);
    tft.fillRect(2, 180, 240, 240, HX8357_BLACK);
    tft.setCursor(2, 180);
    tft.setTextColor(HX8357_WHITE);
    tft.setTextSize(2);
    tft.print("Por favor, pon los dispositivos juntos y toca la pantalla hasta que se pida otra cosa");

    delay(4000);
    TS_Point p = ts.getPoint();

// El siguiente bucle detecta si la pantalla se está tocando

   if (p.z > MINPRESSURE && p.z < MAXPRESSURE){

      Serial.print("OK");
      tft.fillRect(200, 60, 300, 300, HX8357_BLACK);
    tft.fillRect(2, 180, 300, 300, HX8357_BLACK);
    tft.setCursor(2, 180);
    tft.setTextColor(HX8357_WHITE);
    tft.setTextSize(2);
    tft.print( "Realiza los movimientos de calibración del magnetómetro y vuelve a poner los dispositivos juntos.");
    delay(100);

     delay(10000);

// Se recoge el valor calibrado y se almacena
```

```
        if (hfemur < 180 && htibia < 180) {
      calibh = hfemur - htibia;
    }
    if (hfemur > 180 || htibia > 180) {
      if (hfemur > 180) {
        hfemur = 360 - hfemur;
        calibh = htibia - hfemur;
      }
      if (htibia > 180) {
       htibia = 360 - htibia;
        calibh = htibia - hfemur;
      }

      if (htibia > 180 && hfemur > 180){
        calibh = hfemur - htibia;
      }
    }


    calibp = abs(ptibia - pfemur);
    if (restp == 359 || restp == -359) {
      calibp = 0;
    }
     delay(100);

// Como ya está calibrado el booleano se cambia a "true"
     calibrated = true;
    tft.fillRect(200, 60, 300, 300, HX8357_BLACK);
    tft.fillRect(2, 180, 300, 300, HX8357_BLACK);
```

```
    }
    else {
     delay(5000);
    }

}

// Cuando ya está calibrado se calculan los ángulos
if (calibrated == true){

      if (hfemur < 180 && htibia < 180) {
      resth = hfemur - htibia;
    }
    if (hfemur > 180 || htibia > 180) {
      if (hfemur > 180) {
        hfemur = 360 - hfemur;
        resth = htibia - hfemur;
      }
      if (htibia > 180) {
       htibia = 360 - htibia;
        resth = htibia - hfemur;
      }
       if (htibia > 180 && hfemur > 180){
        resth = hfemur - htibia;
      }
    }

    restp = abs(ptibia - pfemur);
    if (restp == 359 || restp == -359) {
      restp = 0;
// Para que sea más preciso se le resta el valor de calibración
//que se había calculado antes por si hay alguna desviación
//y se muestra el nuevo valor en la pantalla

  int newh = resth - calibh;
  int newp = restp - calibp;
    delay(1000);
    tft.fillRect(200, 60, 240, 200, HX8357_BLACK);
    tft.setCursor(200, 60);
    tft.setTextColor(HX8357_WHITE);
    tft.setTextSize(3);
    tft.print(newh); tft.print( " grados");
    delay(100);
    tft.fillRect(200, 120, 240, 200, HX8357_BLACK);
    tft.setCursor(200, 120);
    tft.setTextColor(HX8357_WHITE);
    tft.setTextSize(3);
    tft.print(newp); tft.print( " grados");
    delay(100);
    delay(500);


}
  }
}
```

```
// Función para conectarse a dispositivos
int findConnHandle(uint16_t conn_handle)
{
  for (int id = 0; id < BLE_CENTRAL_MAX_CONN; id++)
  {
    if (conn_handle == prphs[id].conn_handle)
    {
      return id;
    }
  }

  return -1;
}


// Función de control del tiempo de parpadeo del LED rojo
void blink_timer_callback(TimerHandle_t xTimerID)
{
  (void) xTimerID;

  static uint8_t count = 0;

  if ( count < 2 * connection_num ) digitalToggle(LED_RED);
  if ( count % 2 && digitalRead(LED_RED)) digitalWrite(LED_RED, LOW);

  count++;
  if (count >= 10) count = 0;
}
```

3. CÓDIGO DEL DISPOSITIVO TIBIAL

```
// Se incluyen las bibliotecas de las diferentes placas y sensores para poder utilizarlas
#include <bluefruit.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>


// Configuración de los servicios de la placa Bluefruit (Arduino)
BLEDis  bledis;
BLEUart bleuart;
BLEBas  blebas;

// Inicialización del sensor BNO055 y tiempo de retraso entre muestras
#define BNO055_SAMPLERATE_DELAY_MS (100)

Adafruit_BNO055 bno = Adafruit_BNO055();


// Inicialización del control de tiempo para el parpadeo del LED rojo de la placa Bluefruit
SoftwareTimer blinkTimer;


// Bloque de inicio
void setup()
{
// Inicializar el Serial para ver los datos en caso de que sea necesario
  #ifndef ESP8266
  while (!Serial);     // will pause Zero, Leonardo, etc until serial console opens
#endif
```

```
// Se incluyen las bibliotecas de las diferentes placas y sensores para poder utilizarlas
#include <bluefruit.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>


// Configuración de los servicios de la placa Bluefruit (Arduino)
BLEDis  bledis;
BLEUart bleuart;
BLEBas  blebas;

// Inicialización del sensor BNO055 y tiempo de retraso entre muestras
#define BNO055_SAMPLERATE_DELAY_MS (100)

Adafruit_BNO055 bno = Adafruit_BNO055();


// Inicialización del control de tiempo para el parpadeo del LED rojo de la placa Bluefruit
SoftwareTimer blinkTimer;


// Bloque de inicio
void setup()
{
// Inicializar el Serial para ver los datos en caso de que sea necesario
  #ifndef ESP8266
  while (!Serial);
#endif
```

```
  Serial.begin(115200);

  // Si el sensor no se inicia se muestra en el serial (si está conectado al ordenador) que no se ha detectado
  if(!bno.begin())
  {
    Serial.print("No se ha detectado el sensor BNO055");
    while(1);
  }

  bno.setExtCrystalUse(true);



  // Llamada a la función que controla el LED e inicio del mismo
  blinkTimer.begin(1000, blink_timer_callback);
  blinkTimer.start();

  // Conectar el LED azul que indica la conexión del BLE (Bluetooth de baja energía)
  Bluefruit.autoConnLed(true);

  // Configuración del ancho de banda de las conexiones periféricas
  Bluefruit.configPrphBandwidth(BANDWIDTH_MAX);

  // Configuración de la placa Bluefruit
  Bluefruit.begin();
  Bluefruit.setTxPower(4);
  // Nombrar el dispositivo para que el dispositivo central lo reconozca
  Bluefruit.setName("Tibia sensor");
  Bluefruit.setConnectCallback(connect_callback);
  Bluefruit.setDisconnectCallback(disconnect_callback);

  // Configurar la información del dispositivo
  bledis.setManufacturer("Adafruit Industries");
  bledis.setModel("Bluefruit Feather52");
  bledis.begin();

  // Configuración e inicio del BLE uart
  bleuart.begin();

  // Inicio del servicio de batería
  blebas.begin();
  blebas.write(100);

  // Llamada a la función para mostrarse a los demás dispositivos
  startAdv();

}


// Función que sirve para mostrar el dispositivo a los demás
void startAdv(void)
{

  Bluefruit.Advertising.addFlags(BLE_GAP_ADV_FLAGS_LE_ONLY_GENERAL_DISC_MODE);
  Bluefruit.Advertising.addTxPower();
  Bluefruit.Advertising.addService(bleuart);
  Bluefruit.ScanResponse.addName();
  Bluefruit.Advertising.restartOnDisconnect(true);
  Bluefruit.Advertising.setInterval(32, 244);    // in unit of 0.625 ms
```

```
  Bluefruit.Advertising.setFastTimeout(30);      // number of seconds in fast mode
  Bluefruit.Advertising.start(0);                // 0 = Don't stop advertising after n seconds
}

// Función principal de repetición
void loop(void) {

   // Se inicializan las diferentes variables
   uint8_t buf[64];
   float pi=3.14159;
   String sheading;
   String spitch;
   String sroll;

   // Recolección de los datos del sensor
   imu::Vector<3> euler = bno.getVector(Adafruit_BNO055::VECTOR_EULER);


  // Obtención de la calibración del sensor. 0 significa no calibrado mientras que 3 significa totalmente calibrado
  uint8_t system, gyro, accel, mag = 0;
  bno.getCalibration(&system, &gyro, &accel, &mag);


  // Muestra los valores en el Serial, en el caso de que sea necesario
  //(Normalmente esto no se utiliza, ya que se visualiza todo finalmente en la pantalla)
  Serial.print("Sys:");
  Serial.print(system, DEC);
  Serial.print(" Gyro=");
  Serial.print(gyro, DEC);
  Serial.print(" Accel=");
  Serial.print(accel, DEC);
  Serial.print(" Mag=");
  Serial.println(mag, DEC);
  Serial.println("");

  // Si el sensor está completamente calibrado se muestra por pantalla un "OK" y además
  // Se obtienen los datos de Heading, Pitch y Roll en formato de punto flotante.
  // Los datos en punto flotante (Float) se pasan a String
  // Cada bit se envía al dispositivo central mediante BLE UART

  if (system == 3 && gyro == 3 && accel == 3 && mag == 3){
    Serial.print ("OK");

      // ------------------------------------------------ HEADING ------------------------------------------------//

    delay(100);
    float heading = euler.x(); // Valor en punto flotante


    sheading = String(heading); // Convertido en String
    uint8_t utf[sizeof(sheading)];

    // Separación en bits
    for(int l=0; l <sizeof(sheading)+1; l++){
      utf[l] = sheading.charAt(l);
    }

    bleuart.write( utf, sizeof(sheading)); // Envío de cada bit
```

```
  // ------------------------------------------------ PITCH ------------------------------------------------/

  float pitch = euler.z();

  pitch = pitch + 1000; // Se suma 1000 para tener una distinción entre las otras  dos variables
  spitch = String(pitch);
  uint8_t vtf[sizeof(spitch)];

  for(int k=0;k<sizeof(spitch)+1; k++){
   vtf[k]=spitch.charAt(k);
  }

  bleuart.write(vtf, sizeof(spitch));

// ------------------------------------------------ ROLL ------------------------------------------------//

  float roll = euler.y();
  if (roll < 0){
    roll=roll+360;
  }
  roll = roll + 2000; // Se suma 2000 para hacer distinción entre las variables

sroll = String(roll);
uint8_t wtf[sizeof(sroll)];

for(int j=0;j<sizeof(sroll+1); j++){
  wtf[j]=sroll.charAt(j);
}

bleuart.write(wtf, sizeof(sroll));
  waitForEvent();
}
}

// Función para conectarse a un dispositivo central
void connect_callback(uint16_t conn_handle)
{
  char central_name[32] = { 0 };
  Bluefruit.Gap.getPeerName(conn_handle, central_name, sizeof(central_name));

  Serial.print("Connected to ");
  Serial.println(central_name);
}

// Función para desconectarse del dispositivo central
void disconnect_callback(uint16_t conn_handle, uint8_t reason)
{
  (void) conn_handle;
  (void) reason;

  Serial.println();
  Serial.println("Disconnected");
}



// Función de control del parpadeo de la luz LED
void blink_timer_callback(TimerHandle_t xTimerID)
{
  (void) xTimerID;
  digitalToggle(LED_RED);
}

void rtos_idle_callback(void)
{
}
```

4. CÓDIGO DEL DISPOSITIVO FEMORAL

```
// Se incluyen las bibliotecas de las diferentes placas y sensores para poder utilizarlas
#include <bluefruit.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>

// Configuración de los servicios de la placa Bluefruit (Arduino)
BLEDis  bledis;
BLEUart bleuart;
BLEBas  blebas;

// Inicialización del sensor BNO055 y tiempo de retraso entre muestras
#define BNO055_SAMPLERATE_DELAY_MS (100)

Adafruit_BNO055 bno = Adafruit_BNO055();


// Inicialización del control de tiempo para el parpadeo del LED rojo de la placa Bluefruit
SoftwareTimer blinkTimer;


// Bloque de inicio
void setup()
{
// Inicializar el Serial para ver los datos en caso de que sea necesario
  #ifndef ESP8266
  while (!Serial);
#endif

  Serial.begin(115200);

 // Si el sensor no se inicia se muestra en el serial (si está conectado al ordenador) que no se ha detectado
  if(!bno.begin())
  {
    Serial.print("No se ha detectado el sensor BNO055");
    while(1);
  }

  bno.setExtCrystalUse(true);


  // Llamada a la función que controla el LED e inicio del mismo
  blinkTimer.begin(1000, blink_timer_callback);
  blinkTimer.start();

  // Conectar el LED azul que indica la conexión del BLE (Bluetooth de baja energía)
  Bluefruit.autoConnLed(true);

  // Configuración del ancho de banda de las conexiones periféricas
  Bluefruit.configPrphBandwidth(BANDWIDTH_MAX);

  // Configuración de la placa Bluefruit
  Bluefruit.begin();
  Bluefruit.setTxPower(4);
  // Nombrar el dispositivo para que el dispositivo central lo reconozca
  Bluefruit.setName("Femur sensor");
  Bluefruit.setConnectCallback(connect_callback);
  Bluefruit.setDisconnectCallback(disconnect_callback);

  // Configurar la información del dispositivo
  bledis.setManufacturer("Adafruit Industries");
```

```
  bledis.setModel("Bluefruit Feather52");
  bledis.begin();

  // Configuración e inicio del BLE uart
  bleuart.begin();

  // Inicio del servicio de batería
  blebas.begin();
  blebas.write(100);

  // Llamada a la función para mostrarse a los demás dispositivos
  startAdv();

}

// Función que sirve para mostrar el dispositivo a los demás
void startAdv(void)
{

  Bluefruit.Advertising.addFlags(BLE_GAP_ADV_FLAGS_LE_ONLY_GENERAL_DISC_MODE);
  Bluefruit.Advertising.addTxPower();
  Bluefruit.Advertising.addService(bleuart);
  Bluefruit.ScanResponse.addName();
  Bluefruit.Advertising.restartOnDisconnect(true);
  Bluefruit.Advertising.setInterval(32, 244);    // in unit of 0.625 ms
  Bluefruit.Advertising.setFastTimeout(30);      // number of seconds in fast mode
  Bluefruit.Advertising.start(0);                // 0 = Don't stop advertising after n seconds
}
// Función principal de repetición
void loop(void) {

   // Se inicializan las diferentes variables
   uint8_t buf[64];
   float pi=3.14159;
   String sheading;
   String spitch;
   String sroll;

   // Recolección de los datos del sensor
   imu::Vector<3> euler = bno.getVector(Adafruit_BNO055::VECTOR_EULER);


  // Obtención de la calibración del sensor. 0 significa no calibrado mientras que 3 significa totalmente calibrado
  uint8_t system, gyro, accel, mag = 0;
  bno.getCalibration(&system, &gyro, &accel, &mag);


  // Muestra los valores en el Serial, en el caso de que sea necesario
  //(Normalmente esto no se utiliza, ya que se visualiza todo finalmente en la pantalla)
  Serial.print("Sys:");
  Serial.print(system, DEC);
  Serial.println("");
  Serial.print(" Gyro=");
  Serial.print(gyro, DEC);
  Serial.print(" Accel=");
  Serial.print(accel, DEC);
  Serial.print(" Mag=");
  Serial.println(mag, DEC);
```

```
// Si el sensor está completamente calibrado se muestra por pantalla un "OK" y además
// Se obtienen los datos de Heading, Pitch y Roll en formato de punto flotante.
// Los datos en punto flotante (Float) se pasan a String
// Cada bit se envía al dispositivo central mediante BLE UART

if (system == 3 && gyro == 3 && accel == 3 && mag == 3){
  Serial.print ("OK");


    // ------------------------------------------------ HEADING ------------------------------------------------//

  delay(100);
  float heading = euler.x(); // Valor en punto flotante

  sheading = String(heading); // Convertido en String
  uint8_t utf[sizeof(sheading)];

  // Separación en bits
  for(int l=0; l <sizeof(sheading)+1; l++){
    utf[l] = sheading.charAt(l);
  }

  bleuart.write( utf, sizeof(sheading)); // Envío de cada bit

  // ------------------------------------------------ PITCH ------------------------------------------------//

  float pitch = euler.z();

  pitch = pitch + 1000; // Se suma 1000 para tener una distinción entre las otras  dos variables

spitch = String(pitch);
uint8_t vtf[sizeof(spitch)];

for(int k=0;k<sizeof(spitch)+1; k++){
  vtf[k]=spitch.charAt(k);
}

bleuart.write(vtf, sizeof(spitch));

// ------------------------------------------------ ROLL ------------------------------------------------//

  float roll = euler.y();

  if (roll < 0){
    roll=roll+360;
  }
  roll = roll + 2000; // Se suma 2000 para hacer distinción entre las variables
sroll = String(roll);
uint8_t wtf[sizeof(sroll)];

for(int j=0;j<sizeof(sroll+1); j++){
  wtf[j]=sroll.charAt(j);
}

bleuart.write(wtf, sizeof(sroll));
```

```
  waitForEvent();
}
}

// Función para conectarse a un dispositivo central
void connect_callback(uint16_t conn_handle)
{
  char central_name[32] = { 0 };
  Bluefruit.Gap.getPeerName(conn_handle, central_name, sizeof(central_name));

  Serial.print("Connected to ");
  Serial.println(central_name);
}

// Función para desconectarse del dispositivo central
void disconnect_callback(uint16_t conn_handle, uint8_t reason)
{
  (void) conn_handle;
  (void) reason;

  Serial.println();
  Serial.println("Disconnected");
}


// Función de control del parpadeo de la luz LED
void blink_timer_callback(TimerHandle_t xTimerID)
{
  (void) xTimerID;
  digitalToggle(LED_RED);
}
```

```
void rtos_idle_callback(void)
{

}
```