UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

TRABAJO FIN DE GRADO

Diseño de PCB para orientación de un robot y sensado de la batería

Alumno: **Jordi Armengol Miralles**

Tutor: Juan Ramón Rufino Valor

GRADO EN INGENIERÍA ELÉCTRICA

Convocatoria de defensa: **septiembre de 2019**

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Resumen

En este documento se detalla el proceso del diseño de una PCB para su posterior colocación en un robot humanoide de la marca bioloid y que éste pueda orientarse debido a los diversos sensores que contiene.

Para el diseño se han elegido los diferentes componentes como reguladores de tensión y un convertidor DC/DC para los diferentes niveles de tensión que se necesitan.

Mediante una IMU se han realizado medidas para que el robot pueda referenciarse en el espacio. Se han obtenido datos a través de la comunicación I2C mediante un microprocesador y se han procesado para su utilización.

Una vez ya se han obtenido datos útiles se han trasmitido por UART o protocolo serie mediante el bus que tiene el robot y así poder comunicarnos con el propio formato de comunicaciones de éste.

Para la programación del microprocesador se ha utilizado el lenguaje de programación C++ y además se ha utilizado el propio software del robot para manejar los datos enviados por el microprocesador.

A través del sensor de intensidad colocado en la placa de circuito impreso y el divisor de tensión se ha podido medir el nivel de tensión de la batería para conocer la carga restante y conocer el estado de la batería referente a la pérdida de capacidad que ocurre con el paso del tiempo.

Finalmente se ha conseguido integrar la PCB con el robot bioloid y que mediante los datos proporcionados por los sensores de la IMU se oriente en el espacio.

1 Introducción

1.1 Objetivos

La realización de este trabajo tiene diversos objetivos:

- El diseño, fabricación y montaje de la PCB.
- Realizar el software para que los sensores de la PCB puedan comunicarse a través de varios protocolo con el robot Bioloid.
- Inicializar los sensores de orientación, en este caso se usa una IMU, que nos permite saber la orientación de tres ejes de cada sensor que lleva, un giroscopio, un acelerómetro y un magnetómetro.
- Procesar la información obtenida para que el software del propio Bioloid maneje los datos.
- Unir todo el software para que el robot pueda orientarse y además se pueda saber el estado de la batería.
- El montaje de la PCB con sus sensores en el robot Bioloid
- El robot debe realizar pruebas que puntúan la correcta orientación

2 Diseño de la PCB

Para el diseño de la PCB se han tenido en cuenta varios factores:

1. La limitación del tamaño debido a que la PCB debe ir montada en un robot humanoide y no debe dificultarle los movimientos.

2. Elementos necesarios para la funcionalidad de la placa.

3. Disposición de los componentes.

2.1 Dimensiones máximas

Para averiguar el tamaño de la placa se ha elegido una posición en la cual el robot humanoide pueda llevarla y no limitar los movimientos, en este caso se ha elegido el pecho del robot como se puede observar en la Figura 1, el cual si no se exceden los límites de dimensiones, no habrá problemas. Las dimensiones finales de la PCB son de 57×60 mm, además se ha diseñado una carcasa en 3D para la protección de esta placa la cual añade 3 mm por cada lado, quedando así unas dimensiones finales de 63×66 mm .

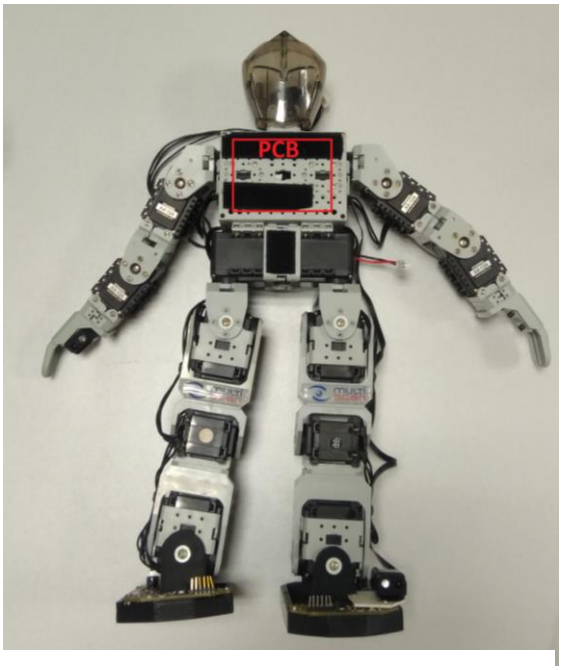


Figura 1: Robot Bioloid

2.2 Elementos de la PCB

Para realizar todas las funciones que se tienen planeadas para la PCB se deben seleccionar una serie de componentes que deben adecuarse y ser compatibles entre ellos.

2.2.1 Sensor de Intensidad

Este sensor se va a utilizar para medir la corriente que circula a través del robot, midiendo la intensidad y el tiempo que está activo el robot, se puede calcular los Ah que consume el robot y comparándolo con la capacidad de la batería se puede averiguar si ésta ya no está en buen estado debido a que su capacidad después de un tiempo ha decaído. Para la elección del sensor se ha tenido en cuenta que el tamaño de este sea pequeño, ligero, la facilidad de uso y que pueda medir corrientes continuas. Teniendo en cuenta los criterios anteriores se descartan los sensores de intensidad basados en bobinas porque son de tamaño más grande, pesado en comparación y solo miden corrientes alternas. Finalmente se ha seleccionado un sensor de corriente de efecto hall, el efecto hall consiste en la aparición de una diferencia de potencial en un conductor o semiconductor por el que circula una corriente al aplicarse un campo magnético en dirección perpendicular a esta. Como la tensión hall depende del producto de la corriente y del campo magnético, resulta sencillo el cálculo de esta corriente con este componente. La medida de la corriente la obtenemos como tensión a la salida. En este caso se ha utilizado el ACS711 como se ve en la Figura 2 debido a su tamaño compacto y a que la alimentación es de 3.3V, que coincide con la del microprocesador e implica un consumo despreciable.

Figura 2: Sensor de intensidad ACS711

2.2.2 Convertidor DC/DC Conmutado

Un Convertidor DC/DC no aislado regulable con resistencias externas con una entrada que puede variar de los 4,5 V a los 14 V basado en una tipología de un Buck con una frecuencia de

conmutación de 600 kHz. Se ha ajustado la salida a 5 V mediante una resistencia de 267Ω, como se recomienda en la hoja de datos del componente. La salida a 5V es necesaria debido a que la comunicación del robot con el microprocesador debe ser a esta tensión y como el microprocesador se alimenta a 3.3 V se necesita una tensión suplementaria.

Este regulador es conmutado para minimizar las pérdidas en el regulador, ya que ha de pasar de 12V, que es la tensión a la que funciona el robot bioloid, a 5 V, y el convertidor tiene una eficiencia energética del 90-93%, mientras que en los reguladores de tensión lineal, cuanto mayor sea la diferencia entre las tensiones de entrada y salida, peor será la eficiencia. La mejora de eficiencia se puede calcular, con respecto a un regulador lineal, que tiene una eficiencia para este caso del 41.67% debido a que tiene que caer una tensión de 7 V , que es más de la mitad de la tensión total, se está consiguiendo una mejora como mínimo del 48.33%. Respecto al precio los convertidores DC/DC tienen un coste un poco superior, pero no lo suficiente como para poner reguladores de tensión lineares. En la Figura 3 se puede observar el convertidor elegido.

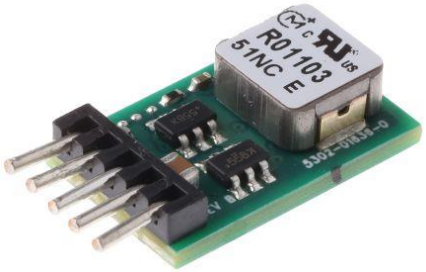


Figura 3 : Convertidor DC/DC OKR-T/3 Series

2.2.3 Reguladores de tensión lineal

Para obtener el nivel de tensión de alimentación de los circuitos internos, se han elegido dos regulador de tensión lineal de tipo fijo de 3V3, uno para la parte digital y otro para la analógica, esto se ha hecho debido a que los circuitos analógicos son más sensibles al ruido que los digitales, y mediante la separación de alimentación y de tierras, se consigue una diferenciación para evitar que la parte digital produzca ruido sobre la analógica. Como el consumo de la parte analógica es muy bajo, se ha elegido un regulador de tensión de menos potencia. En este caso no recurrimos a un convertidor DC/DC debido a que solo perdemos 1.7 V. Se puede observar el regulador de la parte digital en la Figura 4 y el de la parte analógica en la Figura 5.

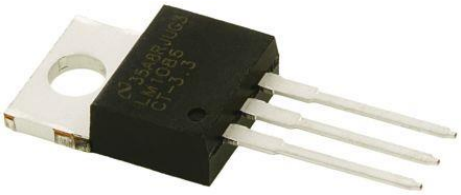


Figura 4: Regulador del circuito digital

Figura 5: Regulador del circuito analógico

2.2.4 Controlador de línea

Se ha elegido un buffer o controlador de línea para permitir el envío y la recepción de datos mediante el protocolo UART del robot, este componente viene recomendado por el fabricante y nos permite activar y desactivar la recepción o el envío de datos. En este caso se ha elegido el SN74LS125ADR como se observa en la Figura 6, que activa las salidas con un control a nivel bajo.

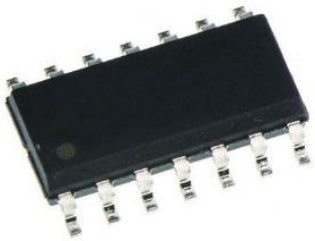


Figura 6: Controlador de línea SN74LS125ADR

2.2.5 Adaptador de niveles de tensión

También se requiere un adaptador de niveles que se utiliza para pasar de 3.3 V a 5 V y viceversa en los protocolos de comunicación serie, ya que el robot se comunica a 5 V y el microprocesador a 3.3 V. Este componente requiere las dos tensión para funcionar, y tiene la capacidad de funcionar a alta frecuencia, que es lo que se busca para las comunicaciones serie. El adaptador elegido es el TXb0104 y se puede observar en la Figura 7 .

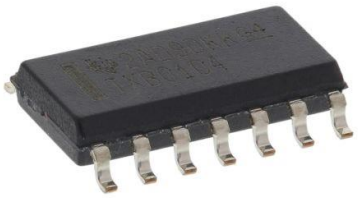


Figura 7: Adaptador de niveles TXb0104

2.2.6 Pulsadores

Se requieren también dos pulsadores para poder seleccionar varias funciones implementadas en el microprocesador . El tamaño de éstos es pequeño debido a la limitación de tamaño.

2.2.7 Conectores de alimentación a 12 V

Se necesitan dos conectores donde por uno vendrá la alimentación de la batería y después de pasar a través del sensor de intensidad, saldrá hacia la CPU del bioloid.

En este caso se ha optado por conectar unos pines macho y unirlos a los cables extensores de la batería, esto nos permite que no haya ningún fallo en la conexión debido a que son distintos los terminales y así también se evita usar más cables extensores. En la Figura 8 se pueden observar.

Figura 8: Cables extensores de la batería del robot bioloid

2.2.8 Conectores de alimentación y datos

También necesitamos dos conectores que se usan en la alimentación y la comunicación serie del robot bioloid. Se necesitan dos conectores debido a que es un bus de datos y si se requiere leer estos, resulta muy sencilla la conexión. Únicamente conectando un convertidor de serie a TTL que viene con el robot bioloid, se pueden recibir datos por el USB del ordenador y visualizarlos con el programa "Terminal". Estos conectores se pueden observar en la Figura 9.

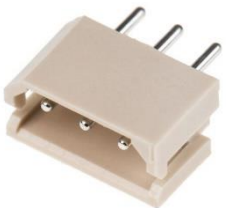


Figura 9: Conector de alimentación del circuito y del bus de datos

2.2.9 Microprocesador

En este caso se ha utilizado un Pic32MX250F180B de la marca microchip como se observa en la Figura 10, este micro permite recoger toda la información de los sensores, procesarlos y enviar por el puerto serie la información si se necesita. Las características importantes de este se encuentran en la Tabla 1.

Figura 10: Micro Pic32MX250F128B

	Características	Necesidad
Comunicación I2C	Sí	Sí
USB	Sí	-
Comunicación UART	Sí	Sí
Entradas Analógicas	Sí	Sí
E/S Digitales	Sí	Sí
Alimentación	3.3V	3.3 V , 5 V

Tabla 1: Características importantes del microprocesador

2.2.10 Conector micro USB

Un conector micro USB como el de la Figura 11 para futuros proyectos que lo requieran, aunque de momento no tiene implementada ninguna función.

Figura 11: Conector micro USB

2.2.11 Ledes

Se ha optado por elegir un led bicolor de montaje superficial de color rojo/verde por la limitación de espacio, como se puede observar en la Figura 12 ,en el cual se han implementado varias funciones como saber el nivel de la batería y la verificación de las comunicaciones.

Figura 12: Led bicolor de montaje superficial

2.2.12 Conector para IMU

También es necesario un conector para una IMU (Unidad de Medida Inercial), la cual lleva un giroscopio de 3 ejes, una magnetómetro de 3 ejes y un acelerómetro de 3 ejes, esto permite el posicionamiento en el espacio muy preciso a través del protocolo de comunicaciones I2C. En este caso se ha optado por una IMU 9 v5 de la empresa Pololu, como se ve en la Figura 13 ,debido a que ya se habían utilizado dispositivos de este tipo y podría resultar más sencillo la implementación de la programación.

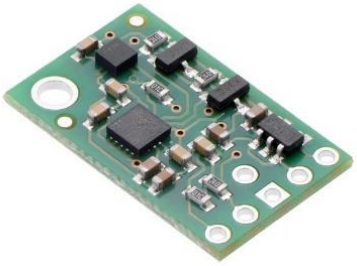


Figura 13: IMU 9 v5

2.2.13 Conector para un Display

El conector permite conectar y desconectar un display para la visualización de funciones. Este display al igual que la IMU está conectado al bus I2C del microprocesador. Se puede observar en la Figura 14.

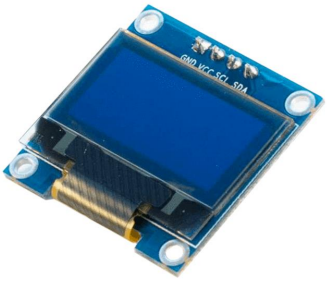


Figura 14: Display I2C

2.2.14 Conector para Pi Kit 3

El conector para el pikit3, Figura 15. Este complemento del microprocesador nos permite programarlo de forma sencilla desde el ordenador mediante el programa “MPLAB X IDE” .

Figura 15: PiKit 3

2.2.15 Componentes pasivos

Se han seleccionado condensadores y resistencias de tipo SMD o de montaje superficial, Figura 17 y Figura 16 respectivamente, mediante los valores recomendados para cada componente.

Figura 17: Condensador SMD

Figura 16: Resistencia SMD

2.3 Disposición de los elementos

Para la disposición de los componentes se ha tenido en cuenta la disposición de la alimentación, que llega a través de la batería en la espalda del robot, pasa a través del sensor de intensidad de efecto Hall, así llegando a la CPU del robot, después se alimenta el bus de datos de la placa, que viene a 12 V, a partir de ahí se introducen los reguladores y demás componentes.

3 Diseño de la PCB en Kicad

Para el diseño de la placa y sus conexiones, se ha utilizado el programa Kicad, que es gratuito y no limita la construcción por serlo. Este permite hacer los esquemas, crear componentes, asignar huellas a estos componentes , situarlos y unirlos por pistas. También nos permite crear los ficheros necesarios para que la PCB sea fabricada con las máquinas preparadas para esta tarea.

A continuación se va a separar el diseño de la PCB en Kicad en tres subapartados:

1. Esquemas de conexiones
2. Disposición de las huellas
3. Archivo de taladrado y de pistas

3.1 Esquemas de conexiones

3.1.1 Sensor de intensidad, divisor de tensión y conectores de alimentación

En este esquema de conexión, Figura 18, se puede observar que también se ha hecho un divisor de tensión para medir el voltaje de la batería a niveles que admite el microprocesador, 3.3 V, el resto de componentes vienen recomendados por el fabricante para que funcione correctamente.

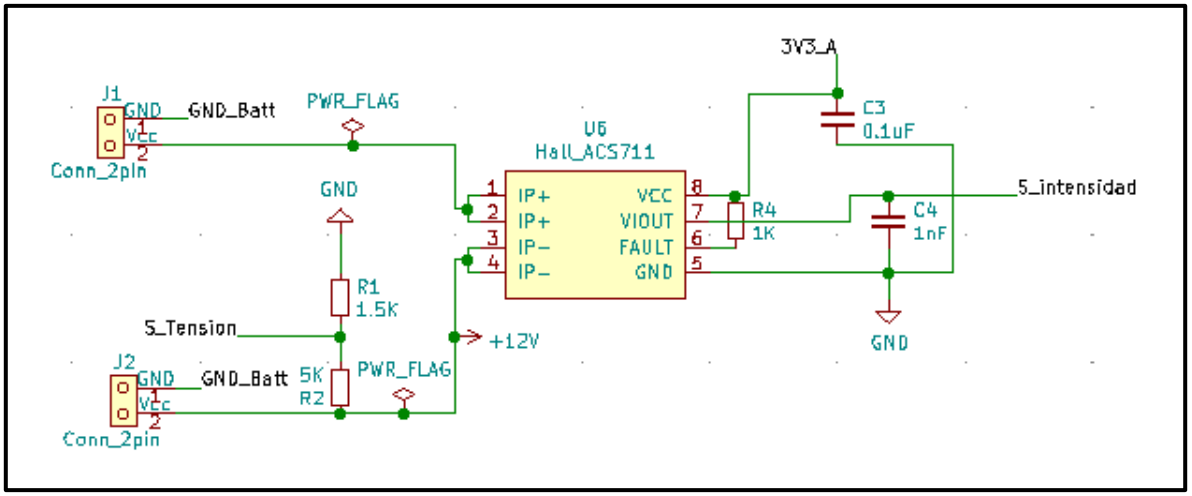


Figura 18: Esquema del sensor de intensidad, divisor de tensión y conectores de alimentación

3.1.2 Convertidor DC/DC y conector de alimentación y datos

En el esquema de conexión del convertidor DC/DC, Figura 19,se puede observar que se ha añadido a la salida una resistencia de 0 Ω, para separar el GND digital “GNDD” y el analógico “GND” , esto se hace debido a que las señales digitales y el propio convertidor generan ruido eléctrico que pueden alterar las señales analógicas a consecuencia de su sensibilidad. También se observan los conectores de alimentación del circuito antes de llegar al convertidor y el bus de datos.

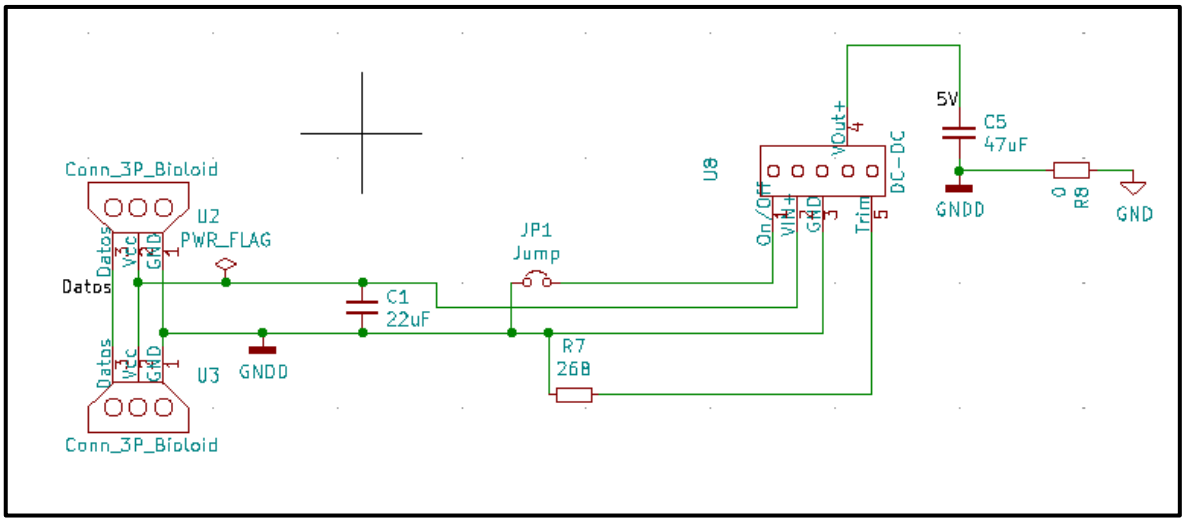


Figura 19: Esquema del convertidor DC/DC y conector de alimentación y datos

3.1.3 Conexiones del microprocesador y Pi kit 3

El esquema de conexiones del microprocesador, Figura 20, muestra la distribución que se ha seguido para conectar todos los componentes y dispositivos. Si observamos el esquema vemos algunos componentes pasivos como por ejemplo en el pin 20, este componente viene recomendado por el fabricante, también observamos la separación de las alimentaciones digital y analógica, en el pin 13 y 28 respectivamente. La alimentación analógica solo alimenta los pines analógicos, evitando así el ruido que se podría generar, como se ha comentado anteriormente. También se observa el conector para el Pi kit 3.

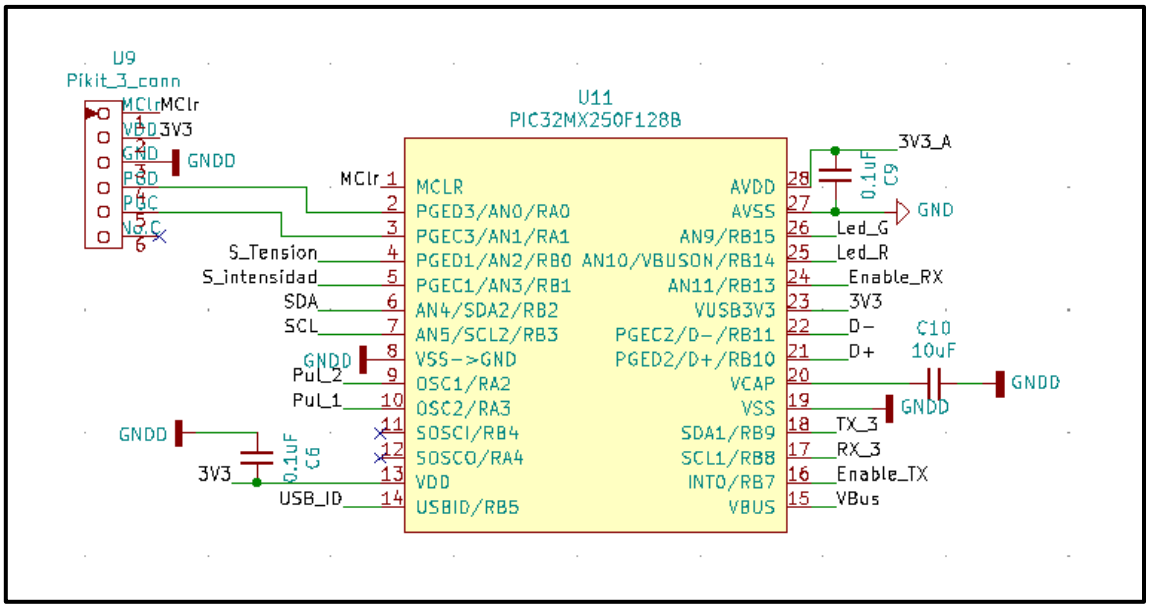


Figura 20: Esquema del microprocesador y pikit 3

3.1.4 Reguladores de tensión lineales

En el esquema de conexiones realizado en la Figura 21, se muestran los valores recomendados de condensadores para los reguladores, de la parte analógica, a la izquierda, y digital, a la derecha.

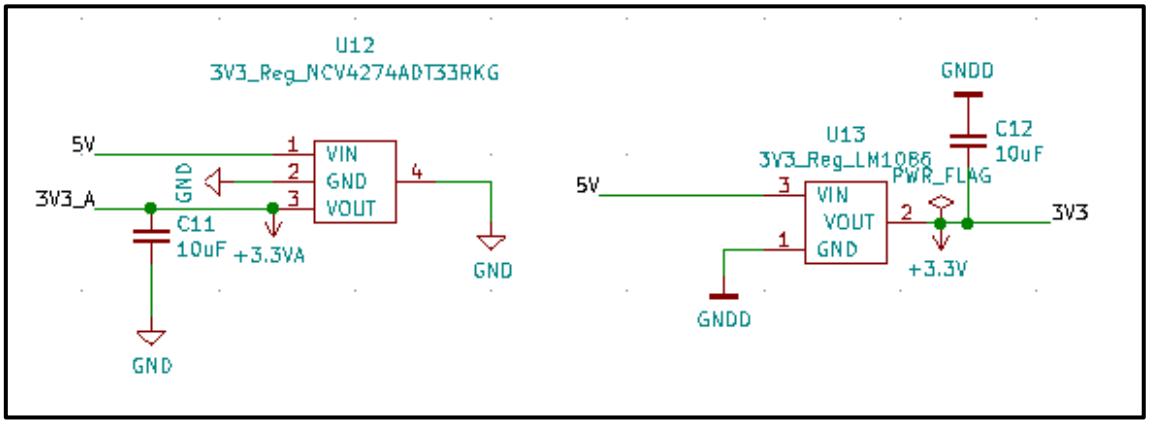


Figura 21: Esquema de los reguladores de tensión lineales

3.1.5 Led bicolor y pulsadores

En el siguiente esquema, Figura 22,s e observa cómo se han añadido unas resistencias de "pull-up" a los pulsadores para que el valor leído por el microprocesador sea nivel alto mientras no se está pulsando y que al pulsarlo y conectar la lectura con GND se lea un nivel bajo, de este modo evitamos estados de incertidumbre. También se han añadido unas resistencias a cada color del led bicolor para limitar la intensidad de estos y que no se estropeen.

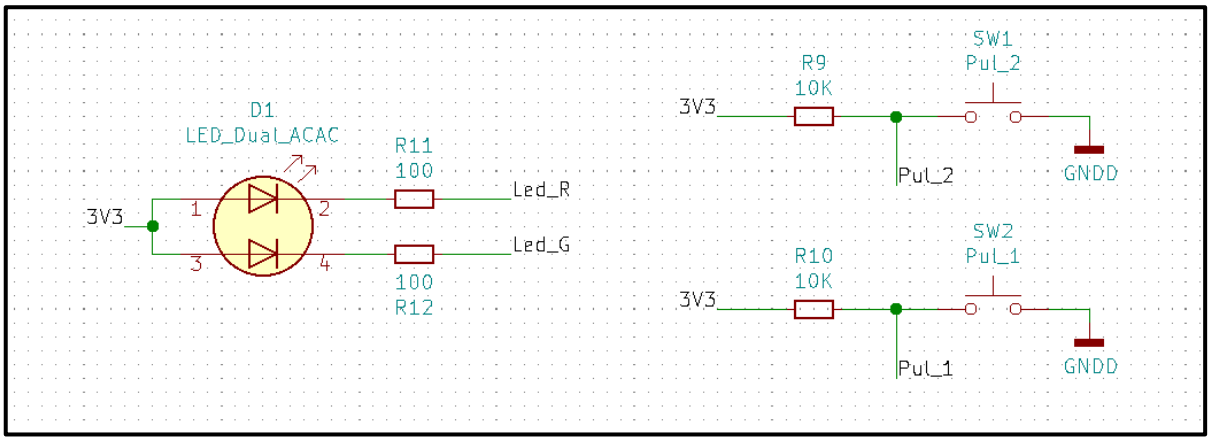


Figura 22: Esquema del Led bicolor y los pulsadores

3.1.6 Buffer de comunicaciones y adaptador de niveles

Para que el microprocesador pueda adquirir datos a 3.3 V y enviarlos a 5 V, se ha añadido un adaptador de niveles de tensión que lo permite, como se puede observar en la Figura 23. También se observa el buffer de comunicaciones para habilitar o no las comunicaciones serie.

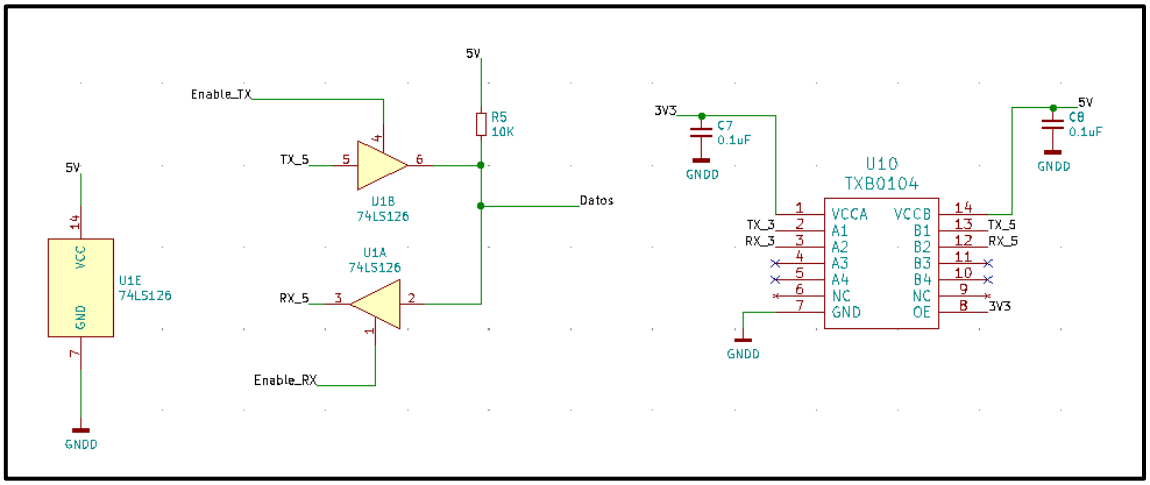


Figura 23: Esquema de adaptador de niveles y buffer de comunicaciones

3.1.7 Conectores de la IMU, el Display y el micro USB

En la Figura 24, se observa el esquema de conexiones de la IMU, el display y el micro USB.

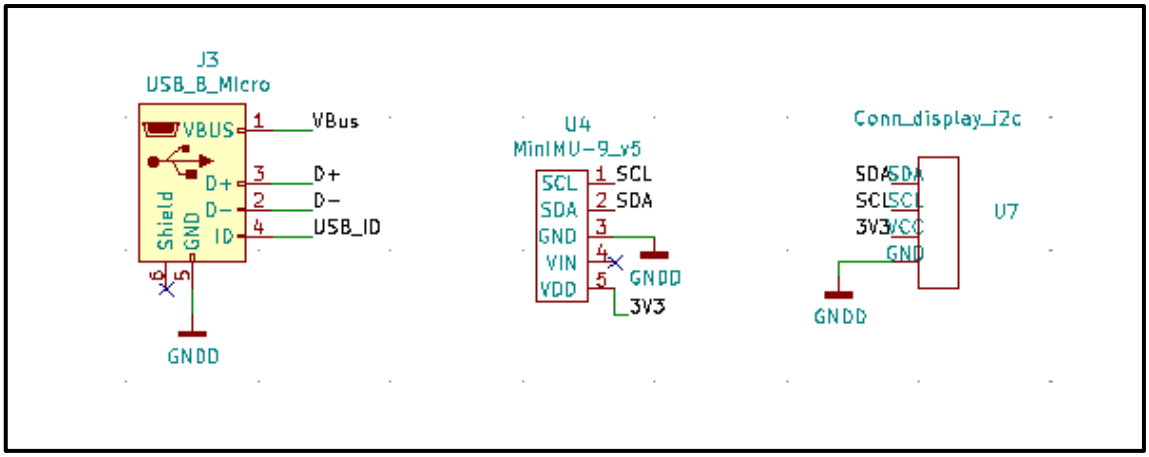


Figura 24: Esquema de los conectores de la IMU, el display y el micro USB

Finalmente en la Figura 25 se muestran las siguientes conexiones, que son elementos necesarios , ya que el bus de comunicaciones I2C necesita un "Pull-up" para funcionar, y el programador del microchip requiere de estos componentes también.

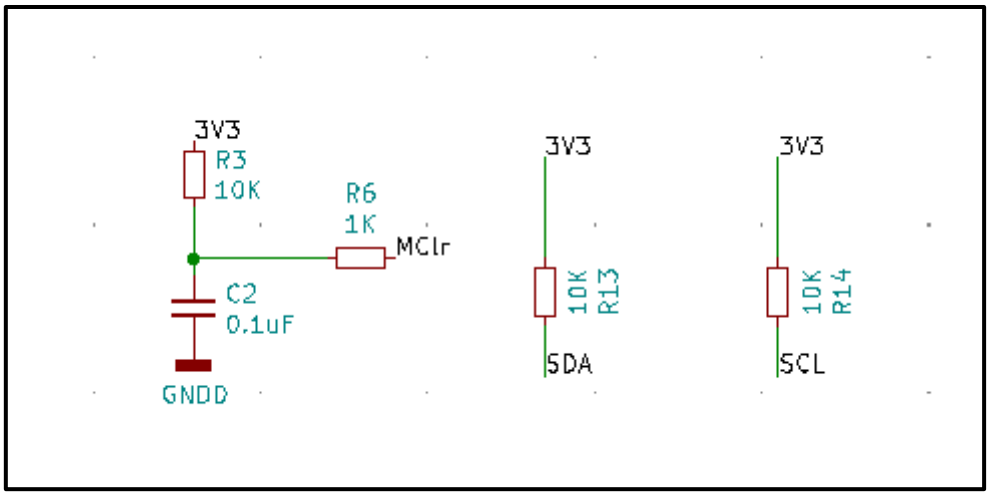


Figura 25: Esquema de elementos para el bus I2C y el conector del Pikit 3

3.2 Disposición de las huellas

Una vez realizadas las conexiones, se ha procedido a diseñar el montaje de los componentes en la PCB mediante las huellas. Las huellas son la forma a tamaño real de los componentes, al haberse estandarizado gran parte de la electrónica, la mayoría de las huellas estaban dentro de la base de datos, exceptuando unas pocas.

Para colocar correctamente los componentes se han tenido unos puntos de referencia:

- Los conectores de la IMU y el display deben ir de tal forma que se puedan poner los dos componentes a la vez.
- El conector micro USB debe ser accesible fácilmente
- El conector del Pi kit 3 debe ser accesible
- Como que el cable de la batería del robot viene por la derecha, el conector de alimentación de 12 V debe estar en este lado
- La alimentación del circuito interior viene por arriba a la derecha, esto nos indica que la posición del conector de alimentación y datos debe pasar por este lado
- Se ha juntado dentro de lo posible la alimentación y la pista de GND para no ocasionar un inductancia residual por el entramado del circuito
- La conexión a los pines se ha hecho de forma perpendicular
- Los giros se han hecho como máximo de 45°
- Las pistas de la alimentación se han hecho de mayor tamaño debido a que circula toda la corriente del robot
- Se ha separado el GND analógico del digital para que el GND analógico no adquiera el ruido del digital

Después de seguir estos pasos el resultado se puede observar en la Figura 27 y la Figura 26.

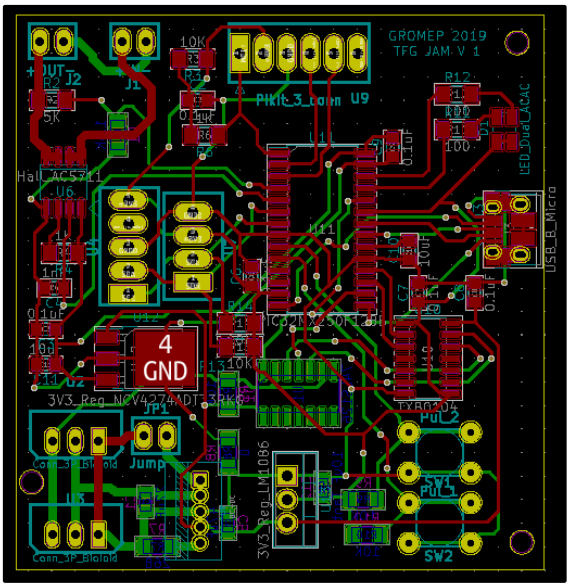


Figura 27: Disposición de las pistas y componentes resaltando la cara frontal

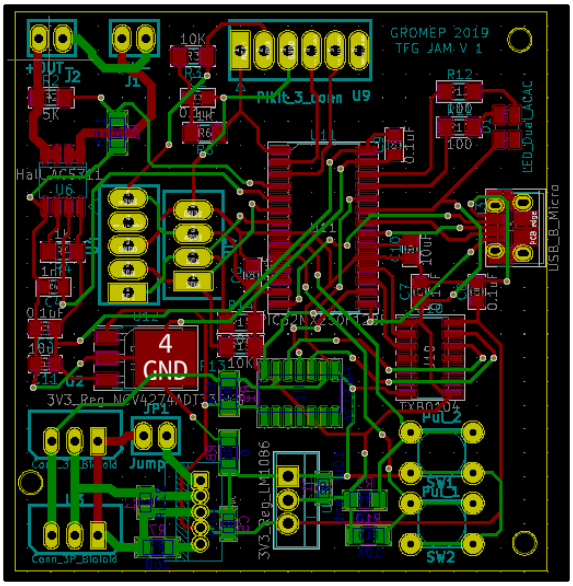


Figura 26: Disposición de las pistas y los componentes resaltando la cara trasera

3.3 Archivo de taladrado y de pistas

Una vez ya está hecha la disposición de los componentes y de las pistas de la PCB, se debe preparar el archivo para que las máquinas automatizadas puedan realizar el dibujo y el taladrado de la placa, para ello se deben seguir los pasos que recomienda el fabricante de PCB y comprimir los archivos generados en un zip, el resultado final se puede observar en la Figura 28.

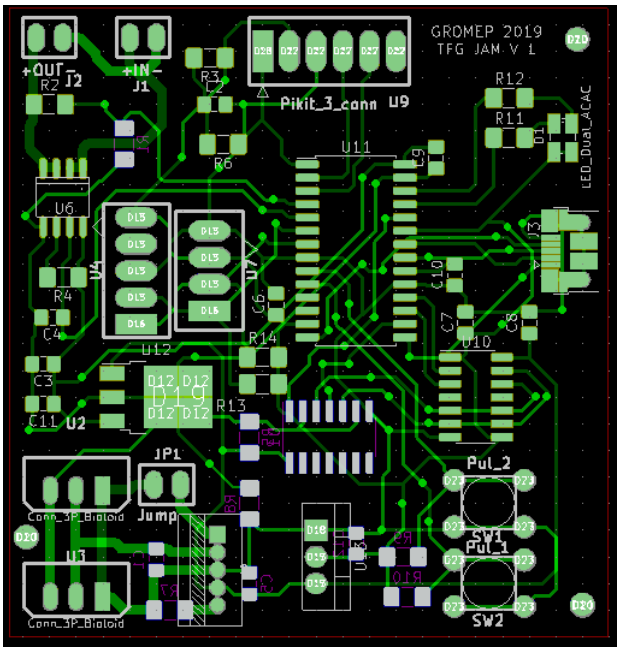


Figura 28: Archivo de taladrado y dibujo de las pistas

3.4 Resultado

El resultado de la fabricación se puede observar en la Figura 30 y la Figura 29.

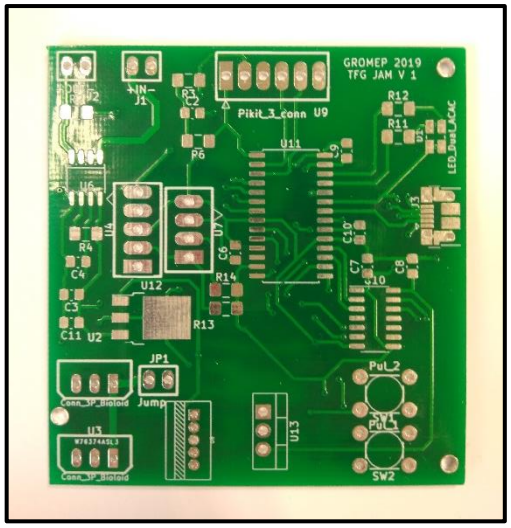


Figura 30: Resultado de la fabricación de la PCB, cara frontal

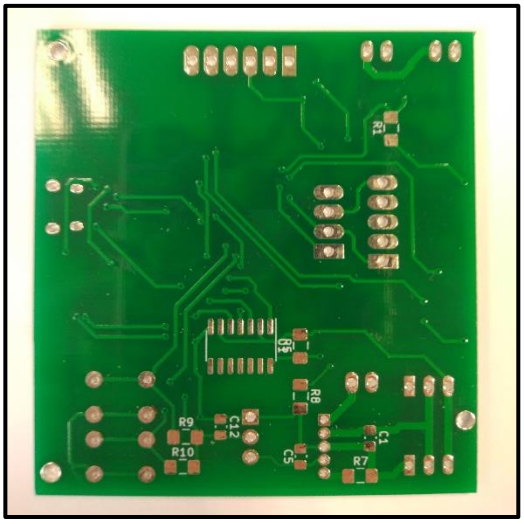


Figura 29: Resultado de la fabricación de la PCB, cara trasera

4 Montaje de la PCB

Para la colocación de los componentes en la PCB se ha procedido a soldarlos, para ello se ha usado una estación de soldadura Weller 3M , como la que se puede ver en la Figura 31.

Figura 31: Estación de soldadura Weller WR 3M

Esta estación de soldadura dispone de tres elementos básicos:

- Un set de un soldador de lápiz WP 120, Figura 32, con el que se realizan las soldaduras de los componentes con agujero pasante y los componentes SMD fáciles de soldar, como las resistencias y los condensadores. Estas soldaduras se ralizan con hilo de estaño y plomo.

Figura 32: Set de soldador tipo lapiz WP 120

- Un set de un soldador de pistola de aire caliente Hap 1, Figura 33, con el que se realizan las soldaduras de los componentes SMD con varias patas, lo que dificulta mucho el soldado tradicional. Para realizar este tipo de soldadura se deben seguir los siguientes pasos:

- Limpiar la zona donde se va a realizar la soldadura
-Aplicar flux, este líquido permite que el estaño se adhiera a las partes metálicas con mucha más facilidad
-Aplicar pasta de soldadura sobre los pines sin poner pasta en exceso
-Colocar el componente sobre la pasta, cerciorándose que esta también sobre los pines correspondientes.
-Calentar la zona con la pistola de aire caliente hasta que todo el estaño se haya adherido al metal de las patas de los componentes y a las pistas

Figura 33: Set de soldador de aire caliente Hap 1

- Un set de un desoldador de vacío DSX 80, Figura 34, el cual no es imprescindible pero agiliza el trabajo de soldar la PCB debido a que si cometes algún error leve puedes retirar el estaño con suma facilidad.

Figura 34: Set de desoldador de vacío DSX 80

4.1 Resultado

El resultado final se puede observar en la Figura 35.

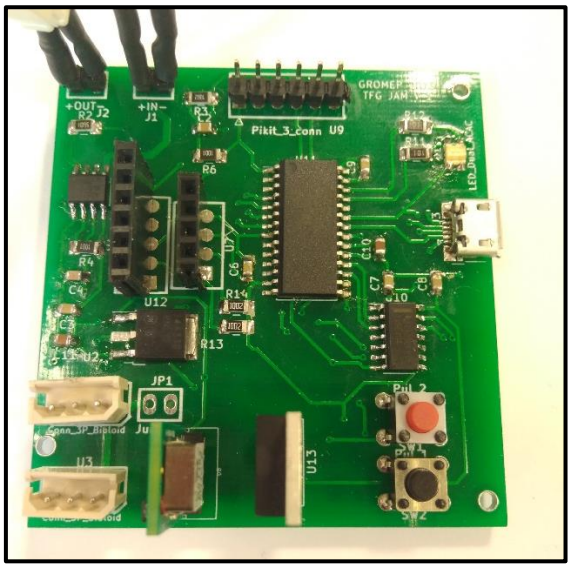


Figura 35: PCB con los componentes montados

5 Comunicaciones

Para que el robot reciba los datos de los sensores de orientación, el sensor primero deberá enviar las medidas al microprocesador por el protocolo de comunicación I2C, ya que este sensor lo utiliza para comunicarse, debido a que es un módulo que da muchos datos, y a continuación, después de procesar la información, deberá enviársela al robot por el protocolo de comunicación serie. A continuación se detallan los métodos y como se pueden aplicar.

5.1 Comunicación I2C

La comunicación I2C (Inter-Integrated circuit Comunications), está implementado en el PIC32 usado en un módulo, este módulo permite la comunicación I2C entre dos o más dispositivos a un alta velocidad.

Para que el bus I2C sea bidireccional se ha implementado un sistema de verificado o "Acknowledge". El sistema de verificado o "ACK", permite a los datos ser enviados a una dirección de un dispositivo por el bus I2C, y éste, contestará un "ACK" para indicar que ha recibido los datos.

I2C es un protocolo síncrono que permite que un dispositivo maestro inicie la comunicación con un dispositivo esclavo. Los datos se intercambian entre estos dispositivos.

El dispositivo maestro controla la línea del reloj, SCL. Esta línea dicta el tiempo de todas las transferencias en el bus I2C. Los dispositivos esclavos pueden manipular esta línea, pero solo pueden forzar el nivel bajo. Esta acción significa que el elemento en el bus no puede tratar con más datos entrantes. Al forzar la línea baja, es imposible registrar más datos en cualquier dispositivo. Esto se conoce como "Clock Stretching".

I2C es una interfaz serie y utiliza solo las dos señales siguientes para intercambiar datos en serie con otro dispositivo:

- SDA: esta señal se conoce como "Serial DAta". Todos los datos enviados de un dispositivo a otro van en esta línea.

- SCL - Esta es la señal del reloj, “Serial CLock”. Lo genera el dispositivo maestro y controla cuándo se envían los datos y cuándo se leen. Como se mencionó anteriormente, la señal se puede forzar a un nivel bajo para que no pueda ocurrir ningún evento.

Las líneas I2C solo pueden tener dos estados eléctricos posibles. Estos estados se conocen como "float high" o nivel alto, Figura 37, y "drive low " o nivel bajo, Figura 36,.El bus I2C necesita resistencias de “pull-up” en las dos líneas, datos y reloj. En reposo el nivel de las líneas es alto, y los dispositivos pueden forzar el nivel bajo. Es por esto por lo que las resistencias “pull-up” son necesarias en I2C.

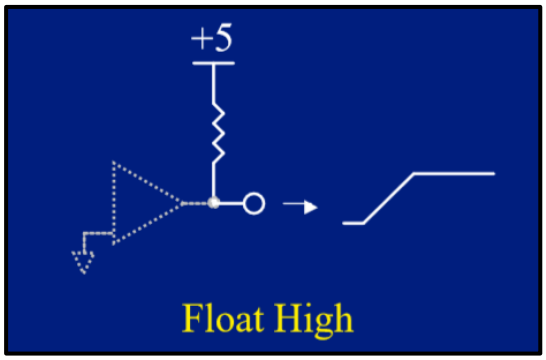


Figura 37: Estado float high o nivel alto

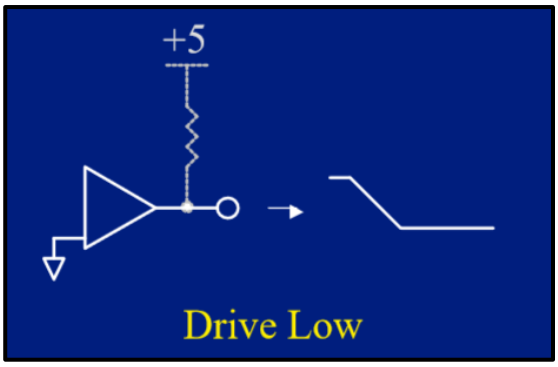


Figura 36: Estado drive low o nivel bajo

El bus I2C tiene una serie de estados. Estos estados indican cuándo se inicia una transferencia, se detiene, se confirma y otros eventos.

Los estados de la señal I2C son:

- Estado de inicio
- Estado de parada
- Estado de reinicio
- Transferencia de datos
- Estado ACK y NACK

A continuación se va a detallar el funcionamiento de cada estado.

5.1.1 Estado de inicio

El estado de inicio indica que un dispositivo desea transferir datos en el bus I2C.

En la Figura 38, se muestra el bloque con una "S" en él, y cómo se ven las señales en el bus I2C. Como se puede observar, SDA primero se establece en nivel bajo, seguido de SCL.

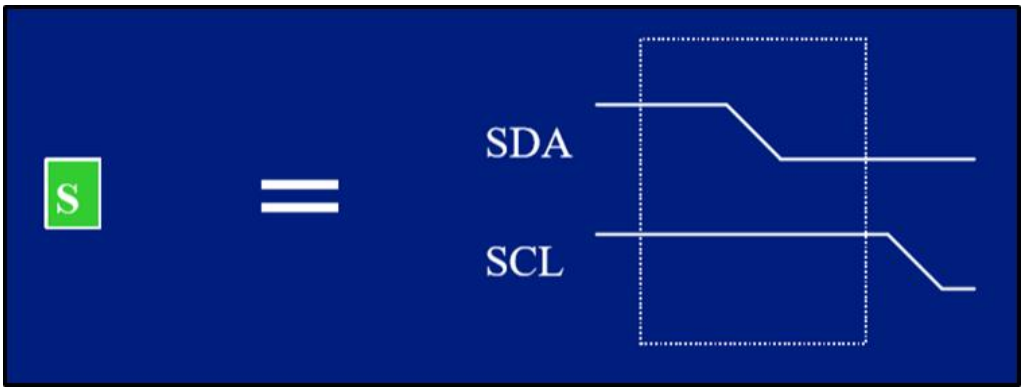


Figura 38: Estado de inicio de comunicación I2C

5.1.2 Estado de parada

Este estado indica que un dispositivo ha finalizado su transferencia en el bus I2C y desea liberar el bus. Una vez liberados, otros dispositivos pueden usar el bus para transmitir datos. Como se puede ver en la Figura 39, un bloque con una "P" representa el estado de parada.

La señalización utilizada para una parada es una puesta de la línea SCL a nivel alto seguido de la línea SDA. Una vez que se completa el estado de parada, tanto SCL como SDA estarán en nivel alto.

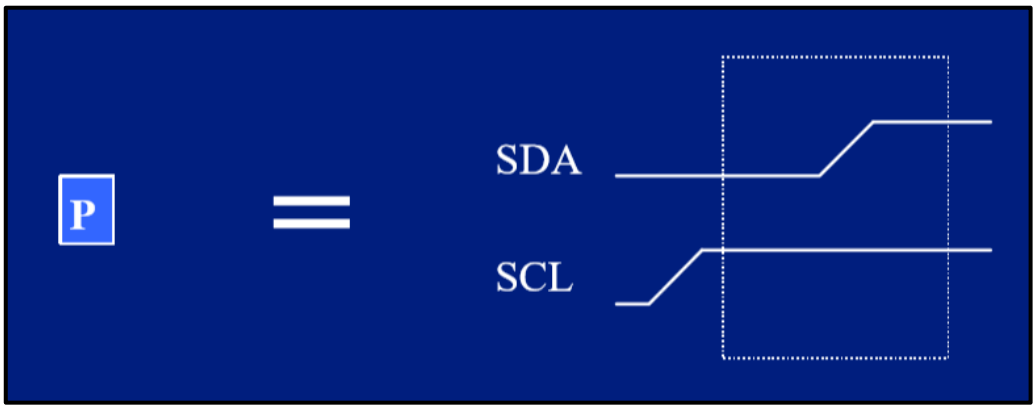


Figura 39: Estado de parada de comunicación I2C

5.1.3 Estado de reinicio

Una estado de reinicio indica que un dispositivo desea transmitir más datos, pero no desea liberar la línea. Esto se hace cuando se debe enviar un inicio, pero no se ha producido una parada. También es una forma conveniente de enviar una parada seguida de un inicio uno después del otro. Evita que otros dispositivos tomen el bus entre transferencias.

El estado de reinicio es representado por una “R” en la Figura 40.

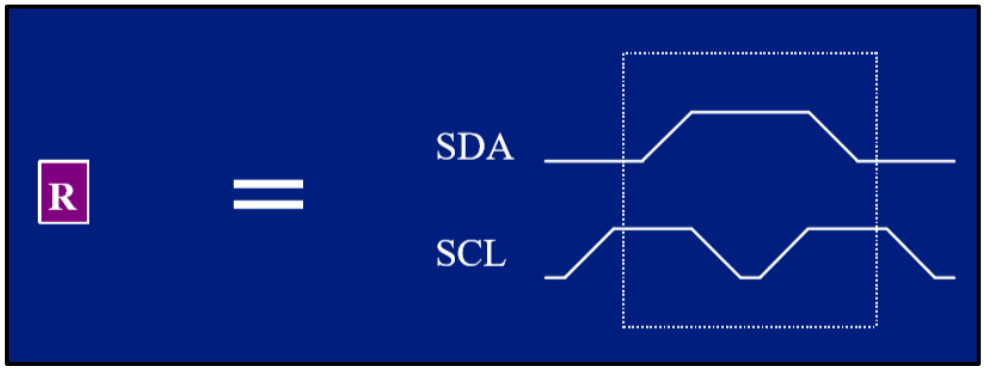


Figura 40: Estado de reinicio de comunicación I2C

La señalización utilizada para un reinicio puede verse como nada más que un estado de parada seguido rápidamente por un estado de inicio.

En la Figura 41 podemos ver claramente lo que se había comentado anteriormente.

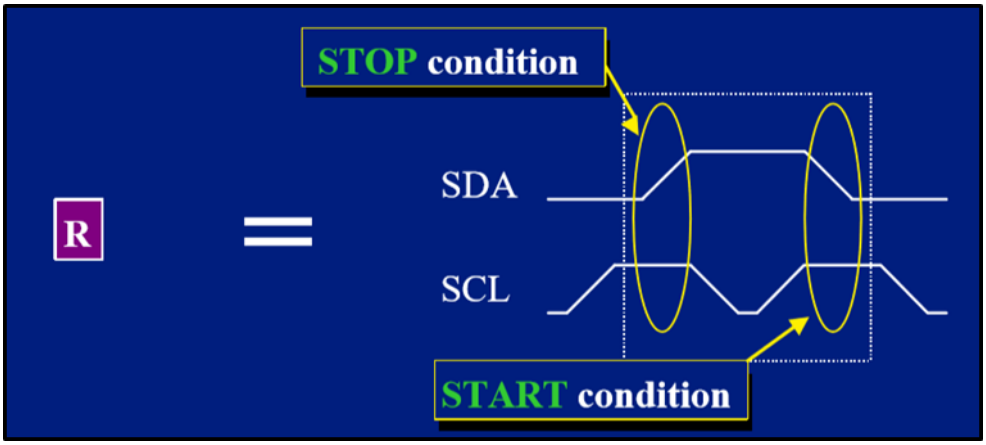


Figura 41: Detalle de estado de reinicio de comunicación I2C

5.1.4 Estado de transferencia de datos

El bloque de datos de la Figura 42 representa la transferencia de 8 bits de información. Los datos se envían en la línea SDA y SCL produce una señal de reloj. El reloj indica cuando se han de leer los datos, indicando si cada bit es un "1" o un "0". Los datos sobre SDA solo se consideran válidos cuando SCL es nivel alto. Cuando SCL no es nivel alto, los datos pueden cambiar. Cuando se comunica a otro dispositivo I2C, los 8 bits de datos pueden ser un código de control, una dirección o datos.

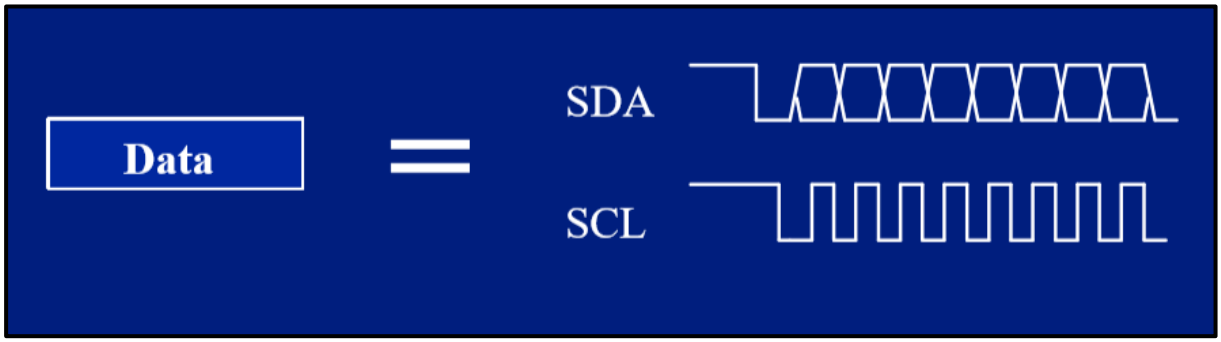


Figura 42: Estado de transferencia de datos de comunicación I2C

5.1.5 Estado ACK y NACK

Un dispositivo puede enviar un bit de "ACK" o confirmar una transferencia de cada byte llevando la línea SDA a nivel bajo durante el noveno pulso de reloj de SCL. Los 9 bits de una transferencia se ven así: 8 bits están sincronizados para los datos, luego, durante el noveno bit, el elemento que recibe los datos toma el bus por un bit. Si controla este bit a nivel bajo, entonces el dispositivo está señalando un "ACK". De lo contrario, si permite que la línea SDA permanezca a nivel alto, está transmitiendo un "NACK". El dispositivo debe bajar el bus activamente para enviar un ACK, pero un NACK podría ser una respuesta pasiva. Este es uno de los beneficios de I2C.

En la Figura 43, se muestra un diagrama que muestra un elemento "ACK". Se observa cómo hay un bloque con una "A" en él.

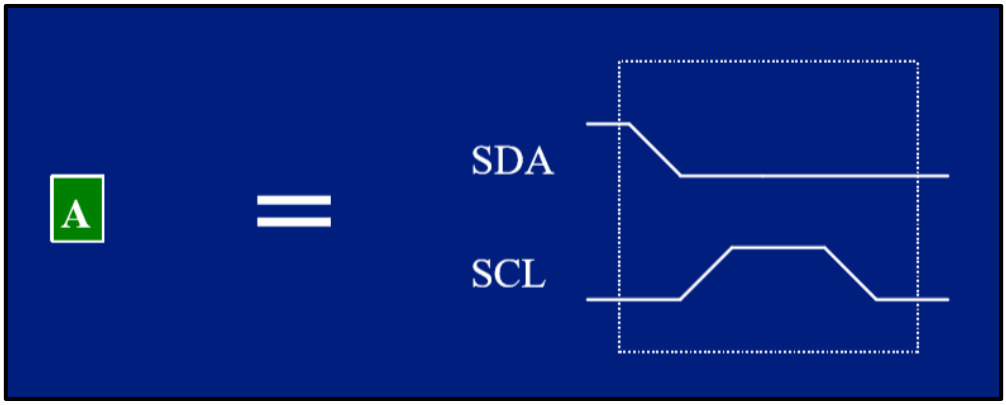


Figura 43: Estado ACK en comunicación I2C

En la Figura 44, se muestra un diagrama que muestra un elemento "NACK". Se observa cómo hay un bloque con una "N" en él.

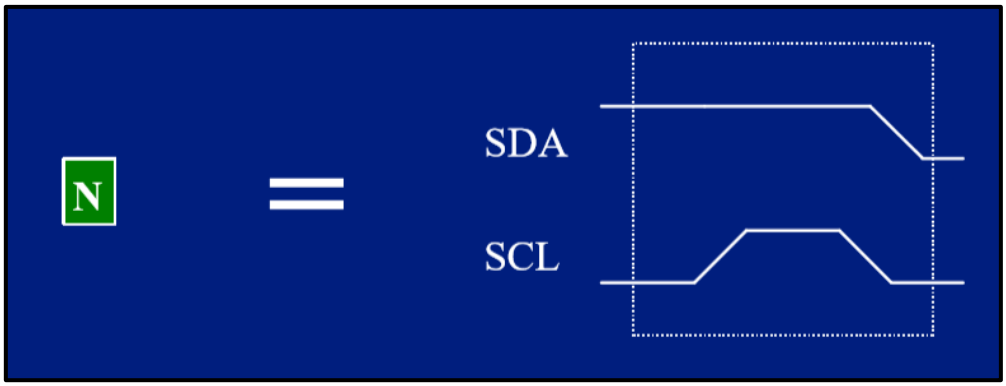


Figura 44: Estado NACK en comunicación I2C

5.1.6 Ejemplo de uso con una EEPROM

Las siglas EEPROM provienen de (*Electrically Erasable Programmable Read-Only Memory*) , esto es un tipo de memoria que puede ser borrada y reprogramada eléctricamente sin ningún método externo, y se suele usar para microprocesadores, aunque actualmente se utilice también las memorias Flash, que son de otro tipo. Este tipo de memoria suele usarse mediante la comunicación I2C.

Si juntamos los elementos mencionados anteriormente, podemos producir transferencias I2C. Necesitamos transferir 3 bytes de información para hacer esto. La transferencia comienza con un inicio, para indicar el comienzo de la transferencia. Luego, se envía el byte de control. El byte de control para una EEPROM puede tener dos bytes de datos diferentes. Una significa que quiere escribir un byte en la EEPROM, y la otra significa que queremos leer un byte de la EEPROM.

La función de escribir en la EEPROM se muestra aquí como "Control IN", que representa poner la EEPROM en un modo de "escritura". Como solo enviamos datos a la EEPROM, usamos el byte "Control In".

A continuación, la EEPROM reconoce este byte. Esto se muestra mediante la "A" después del byte. Se coloca en la siguiente línea para indicar que se transmite por la EEPROM, no por el dispositivo PICmicro.

A continuación el PICmicro envía el byte de dirección. El byte de dirección contiene la dirección de la ubicación de la EEPROM en la que queremos escribir los datos. Dado que la dirección es válida, los datos son aceptados por la EEPROM.

Finalmente, enviamos los datos que queremos escribir. Los datos son entonces validados por la EEPROM. Cuando eso termina, enviamos una condición de parada para completar la transferencia. La parada se representa como el bloque "P" en el extremo. Una vez que la EEPROM obtiene la condición de parada, comenzará a escribir en su memoria. La escritura no se producirá hasta que reciba la condición de parada.

A continuación en la Figura 45 se muestra un ejemplo de escritura.

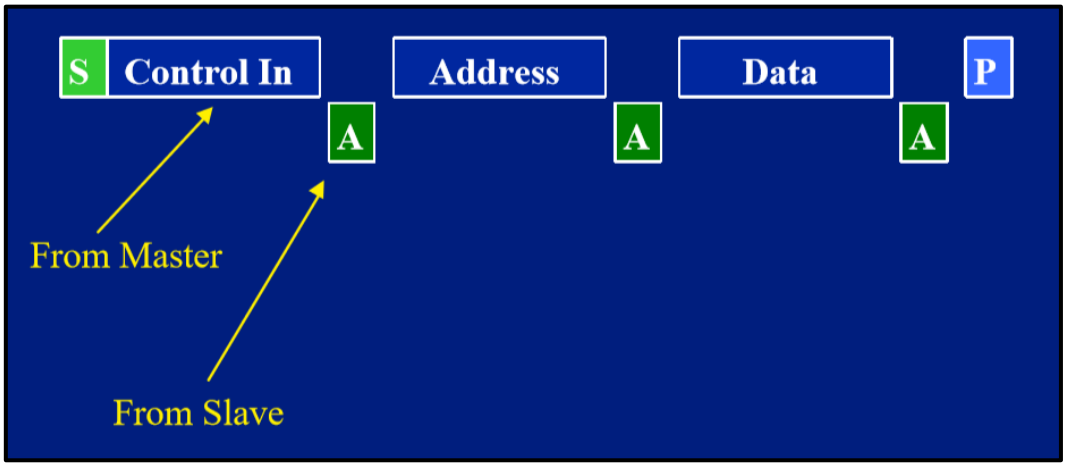


Figura 45: Ejemplo de escritura en EEPROM mediante comunicación I2C

5.2 Comunicación I2C aplicada al microprocesador pic32

Una vez comprendido el funcionamiento básico del I2C pasamos a aplicarlo en el pic32, lo primero que se debe hacer es inicializar las funciones del protocolo para este microprocesador, para ello, el fabricante nos ofrece un manual en el cual se pueden consultar las instrucciones para realizar la inicialización.

5.2.1 Estado de inicio en comunicación I2C del Pic32

A continuación se deben consultar los registros para iniciar la comunicación y esperar a que sucedan unos eventos, como se puede observar en la Figura 46.

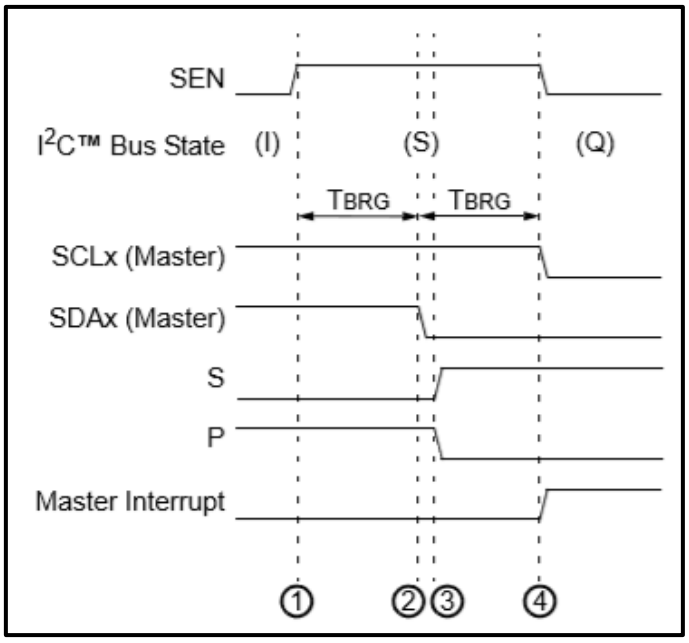


Figura 46: Estado de inicio en comunicación I2C del Pic32

1. En primer lugar se debe establecer el registro "SEN" en nivel alto, lo que produce un evento de inicio en el dispositivo maestro.
2. Después del tiempo establecido el maestro establece SDA en nivel bajo y se espera otra vez un tiempo.
3. Los módulos esclavos detectan la condición de inicio y se establece el registro "S" en 1 y el "P" en 0.
4. Después del tiempo establecido el maestro lleva SCL a 0 y se limpia el registro "SEN". Para que todo esto ocurra, en el programa se deberá esperar hasta que "SEN" sea restablecido a 0.

5.2.2 Estado de transmisión de datos en I2C del Pic32

En la Figura 47 se detalla el proceso para transmitir como dispositivo maestro

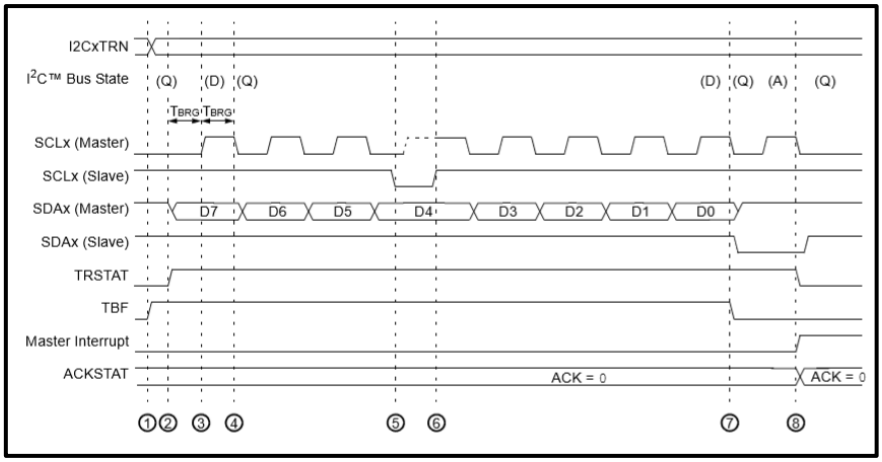


Figura 47: Transmisión de datos con Pic32 en comunicación I2C

1. Al escribir en el registro I2C2TRN, se inicia el evento de transmisión del maestro. Como se puede observar, el registro "TBF" se establece en nivel alto, lo que significa, que está en proceso la transmisión de 8 bits de datos.
2. El control de SDA pasa al registro I2C2TRN. SCL permanece en nivel bajo. El registro que referencia la transmisión de datos por el maestro, TRSTAT , se establece a nivel alto.
3. Se inicia un pulso en el bus SCL para verificar los datos de SDA.
4. Al transcurrir el tiempo establecido, SDA pasa a 0. Después de detectar este nivel bajo el registro I2C2TRN envía el siguiente bit de datos.
5. Si se configura el micro, el dispositivo esclavo tiene la capacidad de establecer la señal SCL a nivel bajo, debido a que no puede recibir más datos, y el maestro se espera sin enviar datos hasta que el dispositivo esclavo libere el bus SDA.
6. El dispositivo esclavo libera SDA y a partir de ahí el maestro continúa transmitiendo.
7. Al terminar des enviar el octavo bit de datos, el maestro libera SDA, manteniéndose así en nivel alto. El registro TBF se limpia y el dispositivo esclavo envía un bit de verificación $\overline{ACK}/NACK$.
8. Después de la caída del noveno pulso de SCL, el dispositivo maestro genera una interrupción. SCL permanece en nivel bajo hasta el siguiente evento. El dispositivo esclavo libera el bus SDA y se limpia el bit TRSTAT.

5.2.3 Estado de recepción de datos en I2C del Pic32

A continuación se muestra el diagrama temporal de la recepción en el dispositivo maestro en la Figura 48.

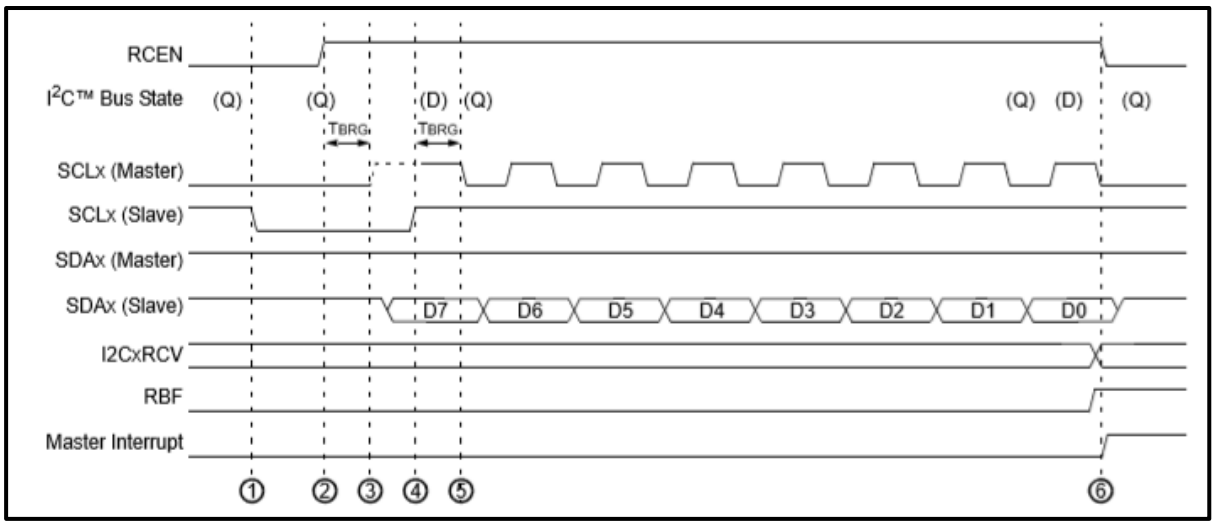


Figura 48:Recepción de datos en comunicación I2C del Pic32

1. El dispositivo esclavo establece la señal de SCL a nivel bajo hasta que esté listo para transmitir datos, cuando libere este bus , se empezarán a transmitir los datos.
2. Estableciendo a nivel alto el registro RCEN, el maestro empezará un evento de recepción de datos. Los pulsos en SCL empezarán después del tiempo configurado, a no ser que el dispositivo este manteniendo la señal SCL a nivel bajo, en este caso esperará hasta que libere el bus ("clock stretch").

3. El dispositivo maestro intenta controlar SCL pero está en el modo de espera.
4. Cuando el dispositivo esclavo libera SCL, se reinician los pulsos en SCL.
5. Al terminar el pulso de SCL, la información de SDA se pasa al registro I2C2RSR, este registro es inaccesible y su funcionamiento es como buffer, es decir, una vez se recibe el dato, este se pasa al registro I2C2RCV, que sí que es accesible.
6. Al terminar el octavo pulso en SCL, se establece el registro RCEN en nivel bajo y el RBF a nivel alto, este registro indica que se ha llenado el registro I2C2RCV, es decir, que se han recibido los datos.

5.2.4 Estado ACK o NACK en I2C del Pic32

Una vez se ha realizado la recepción, el maestro debe enviar un bit de confirmación y así el dispositivo esclavo puede continuar enviando datos o pasando a otro evento. En la Figura 49 se puede observar cómo se realiza esto.

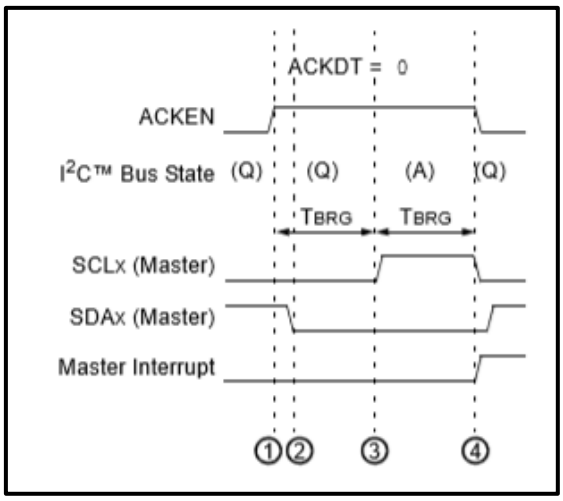


Figura 49: Envió de ACK en I2C del Pic32

1. Estableciendo el registro ACKDT = 0 , se especifica que se va a enviar un bit de verificación $\overline{ACK}$, a continuación se establece el registro ACKEN = 1 para iniciar el evento de verificación del maestro.
2. Al detectar que SCL está a nivel bajo, SDA se sitúa a nivel bajo.
3. Al pasar el tiempo configurado, el generador de pulsos se activa en SCL.
4. Al terminar el pulso se comprueba el estado de SDA para confirmar la verificación del maestro. El registro ACKEN se limpia y se genera una interrupción en el dispositivo maestro.

En la Figura 50, vemos el caso contrario, done el microprocesador deja la línea, para trasmitir un NACK.

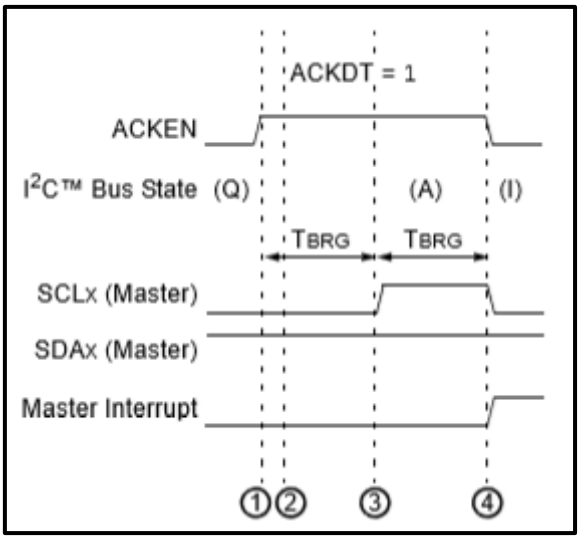


Figura 50:Envio de NACK en I2C del Pic32

1. Estableciendo el registro ACKDT = 1 , se especifica que se va a enviar un bit de verificación NACK, a continuación se establece el registro ACKEN = 1 para iniciar el evento de verificación del maestro.
2. Al detectar que SCL está a nivel bajo, SDA se libera dejándolo a nivel alto.
3. Al pasar el tiempo configurado, el generador de pulsos se activa en SCL.
4. Al terminar el pulso se comprueba el estado de SDA para confirmar la verificación del maestro. El registro ACKEN se limpia y se genera una interrupción en el dispositivo maestro.

5.2.5 Estado de parada en I2C del Pic32

Para finalizar en la Figura 51se detalla el final de la comunicación mediante I2C.

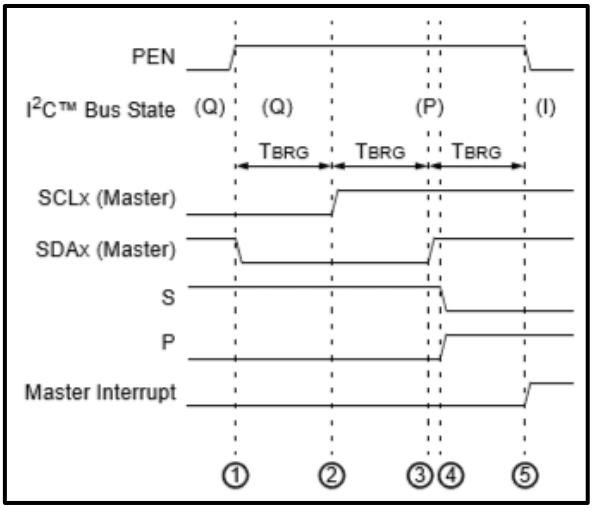


Figura 51: Estado de parada en I2C del Pic32

1. Al establecer el registro PEN = 1, se inicia el evento de parada en el dispositivo maestro. Se empiezan a generar pulsos y el maestro establece SDA a nivel bajo.
2. El maestro libera SCL después de pasar el tiempo entre pulsos.
3. Al pasar el tiempo ente pulsos se libera también SDA.
4. La lógica del dispositivo esclavo detecta la parada. El registro P se establece a nivel alto y la S a nivel bajo.
5. Se limpia el registro PEN y se genera una interrupción en el dispositivo maestro.

5.3 Comunicación I2C de la IMU

Una vez ya está claro el funcionamiento del bus I2C y de la aplicación de este en el Pic32, se debe conseguir comunicarse con la IMU y que nos envíe los datos de su giroscopio, acelerómetro y magnetómetro, para ello se debe conocer los componentes. En el caso del magnetómetro, este componente es un "LIS3MDL", y un "LSM6DS33" para el giroscopio y el acelerómetro. Si buscamos en las hojas de datos de los componentes, se pueden observar que hay diversos registros sobre los que leer o escribir. En primer lugar se debe saber el modo de actuación sobre estos componentes, para ello hay 4 instrucciones.

5.3.1 Escribir un byte al componente esclavo

En la Tabla 2 , se puede observar que el dispositivo maestro, en nuestro caso el Pic32, envía un estado de inicio, a continuación a través de un estado de transferencia de datos, selección la dirección del componente y añade un "0" para escribir o un "1" para leer. Al recibir un estado de ACK del dispositivo esclavo se envía otro estado de transferencia de datos con la dirección del registro en la que se pretende escribir, más tarde vuelve a confirmar la recepción de la información el dispositivo esclavo y se le envían los datos que se desean. Finalmente se finaliza la comunicación mediante un estado de parada después de recibir otro estado de ACK.

Tabla 2: Escritura de 1 byte de datos en el dispositivo esclavo para la IMU

Maestro	Inicio	Dirección + E		Dirección_Reg		DATOS		Parada
Esclavo			ACK		ACK		ACK	

5.3.2 Escribir múltiples bytes al componente esclavo

Para realizar una escritura simultánea de varios bytes sin tener que terminar la comunicación y volver a realizarla, estos componentes nos permiten escribir varios bytes simultáneamente. Para poder hacer esto deberemos indicar al componente que la dirección del registro se vaya autoincrementando para escribir en los registros adyacentes. Para indicar que se quiere un

autoincremento se debe añadir a la dirección del registro un “1” en binario al bit más significativo, es decir un “0b10000000”.

El inicio de la comunicación será igual que en la Tabla 2, pero añadiendo los cambios comentados, y para poder continuar enviando datos, no se deberá terminar la comunicación con un estado de parada, sino que se enviarán más datos, como se puede observar en la Tabla 3.

Tabla 3: Escritura de múltiples datos en la IMU

Maestro	Inicio	Dir		Dirección_Reg + 0b10000000		D		D		D		Parada
Esclavo			A		A		A		A		A	

5.3.3 Leer un byte del componente esclavo

Para realizar la lectura de los registros del componente se realiza la misma secuencia de envío y recepción de datos hasta después de enviar la dirección del registro, en este caso se realiza un reinicio y se envía la dirección del componente añadiendo un “1” de lectura en la parte menos significativa del byte. A continuación el componente envía los datos y el Pic32 responde de forma pasiva con un NACK, como se puede observar en la Tabla 4.

Tabla 4: Recepción de un dato de la IMU

Maestro	Inicio	Dir + E		Dir_Reg		Reinicio	Dir + L			NACK	Parada
Esclavo			A		A			A	Datos		

5.3.4 Leer múltiples bytes del componente esclavo

Para realizar esta función se debe configurar un autoincremento en la dirección del regulador, al igual que se ha hecho anteriormente, y para continuar recibiendo datos no se debe enviar un NACK, sino un ACK hasta que no se quieran más datos, entonces se dejará de enviar, como se puede observar en la Tabla 5.

Tabla 5: Recepción de múltiples datos de la IMU

Maestro	Inicio	Dir + E		Dir_Reg + 0b10000000		Reinicio	Dir + L			A		NA	Parada
Esclavo			A		A			A	D		D		

5.4 Comunicación UART

La comunicación UART (Universal Asynchronous Reciever Transmiter), se usará para la comunicación directa con el robot, ya que este tiene un bus de datos con este protocolo de comunicaciones.

Como no hay señal de reloj en funcionamiento asíncrono, se puede usar un pin para la transmisión y otro pin se pueden utilizar para la recepción. Tanto la transmisión como la recepción puede ocurrir al mismo tiempo, lo que se conoce como operación "full dúplex ".

El UART se puede configurar para transmitir ocho o nueve bits de datos. Si deben transmitirse nueve bits, el noveno bit de datos debe ser colocado en el bit TX9D del registro TXSTA antes de escribir los otros ocho bits en el registro TXREG. Una vez que los datos se han escrito en TXREG, los ocho o nueve bits se mueven en el registro de cambio de transmisión. A partir de ahí son marcados en el pin de TX (Transmit) precedido por un bit de inicio y seguido por un bit de parada.

El uso de un registro de desplazamiento de transmisión separado permite escribir nuevos datos en el registro TXREG mientras se están transmitiendo los datos anteriores, como se observa en la Figura 52.

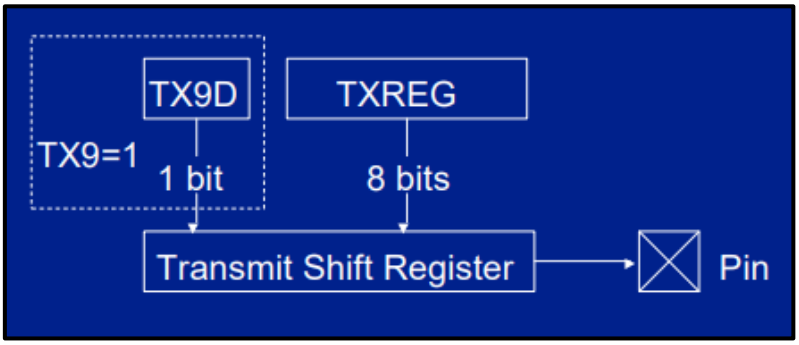


Figura 52: Esquema de envío en UART

El UART se puede configurar para recibir ocho o nueve bits por el bit RX9 en el registro RCSTA.

Después de la detección de un bit de inicio, ocho o nueve bits de datos en serie son pasados del pin RX al registro de cambio de recepción de bit en bit. Después de que el último bit se haya pasado, se comprueba el bit de parada y se pasan los datos al búfer, que los envía por el registro RCREG si está vacío, Figura 53.

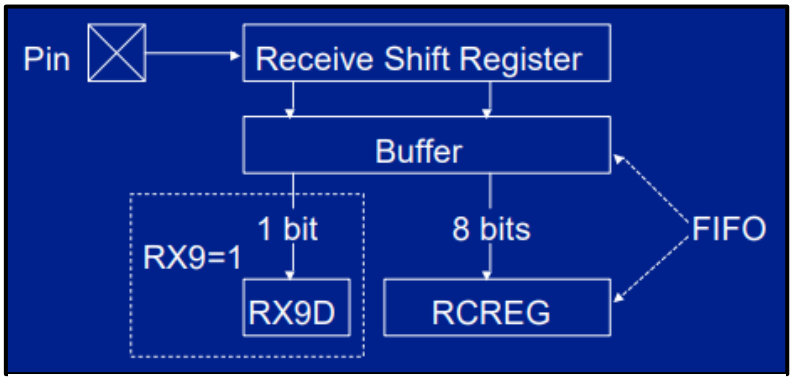


Figura 53: Esquema de recepción en UART

Un ejemplo de transmisión en UART se puede apreciar en la Figura 54.

Figura 54: Ejemplo de transmisión en UART

5.5 UART aplicado al Pic32

Una vez ya se tienen los conceptos básicos de la comunicación por el protocolo UART, se puede aplicar al microprocesador.

El primer paso, al igual que con la comunicación I2c es inicializar este protocolo en el PIC32 mediante unas instrucciones proporcionadas por el fabricante, al hacer esto se debe elegir la velocidad de transmisión con este protocolo, que en este caso nos adaptaremos a la velocidad a la que va el bus del robot, 1 Mbps.

La instrucción para ajustar la velocidad es U2BRG, y para saber cómo asignar la velocidad deseada, se proporciona una ecuación.

$$UxBRG = \frac{F_{PB}}{4 \cdot Baud\ rate} - 1$$

Donde :

- F_{PB}: Velocidad de funcionamiento del microprocesador. En este caso 40MHz.
- "Baud rate" : Velocidad de funcionamiento de UART deseada. En nuestro caso 1Mbps.

$$U2BRG = \frac{40 \cdot 10^6}{4 \cdot 1 \cdot 10^6} - 1 = 9$$

Con esto ya tendríamos iniciado el protocolo UART y ya podríamos enviar y recibir información a través de este.

6 Envío y recepción de información del robot Bioloid

Para comunicarnos con el robot, se debe respetar la estructura de comunicaciones con el que funcionan los motores Dynamixel , ya que nos vamos a comunicar por este bus.

Dynamixel funciona por la recepción de datos binarios. El controlador principal y Dynamixel se comunican entre sí enviando y recibiendo datos llamados paquetes. Hay dos tipos de paquetes: el paquete de instrucciones, que el controlador principal envía para controlar a los Dynamixel, y el paquete de estado, donde Dynamixel responde al controlador principal.

La ID es un número específico para la distinción de cada Dynamixel cuando varios Dynamixels están vinculados a un bus. Al proporcionar la ID a los paquetes de instrucción y estado, el controlador principal puede controlar solo el Dynamixel que desea controlar.

El protocolo usado por Dynamixel es la comunicación serie asíncrona con 8 bits de datos, 1 bit de parada y sin paridad.

Para comunicarnos también tenemos de tener en cuenta que la comunicación con Dynamixel es “half duplex”, la UART “half duplex” o semidúplex es un protocolo de comunicación en serie en el que TxD (Pin de transmisión de datos) y RxD (Pin de recepción de datos) no se pueden usar al mismo tiempo. Este método generalmente se usa cuando muchos dispositivos necesitan conectarse a un solo bus, como es el caso de los motores Dynamixel.

La comunicación tiene un tiempo de retardo entre bytes al enviar un paquete de instrucciones, Figura 55. Si el tiempo de retardo es superior a 100 ms, entonces el actuador Dynamixel lo reconoce como un problema de comunicación y espera de nuevo el siguiente encabezado (0xff 0xff) de un paquete.

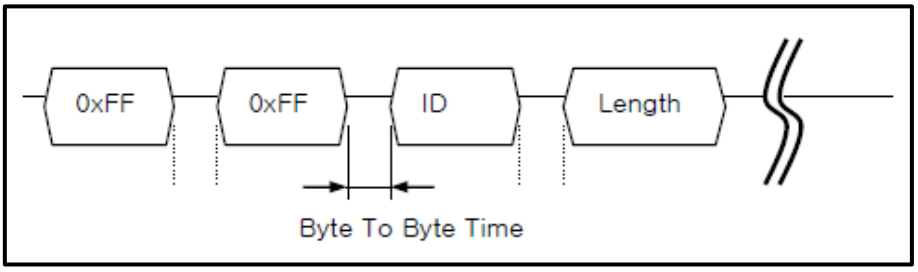


Figura 55: Tiempo entre bytes de la comunicación Bioloid

6.1 Paquete de instrucción

El paquete de instrucciones son los datos de comando enviados al dispositivo, en la Tabla 6 se muestra el paquete de envío general.

Tabla 6: Paquete de instrucción general para bioloid

Encabezdo1	Encabezdo2	ID	Longitud	Instrucción	Param 1	…	Param N	Checksum
0xFF	0xFF	ID	Longitud	Instrucción	Param 1	…	Param N	CHKSUM

A continuación se va a explicar con más detalle cada campo del paquete de instrucción.

6.1.1 Encabezado

Este campo indica el inicio del paquete.

6.1.2 ID

Este campo indica la ID del dispositivo que debe recibir el paquete de instrucciones y procesarlo.

- Rango: 0 - 253 (0x00 - 0xFD), que es un total de 254 números que se pueden usar.
- Broadcast ID: 254 (0xFE), que hace que todos los dispositivos conectados ejecuten el paquete de instrucciones.

6.1.3 Longitud

La longitud del paquete se calcula sumando la Instrucción, Parámetros y Checksum, ya que lo único que puede cambiar son el número de parámetros, podemos definir:

$$longitud = número\ de\ parámetros + 2$$

6.1.4 Instrucción

Este campo define el tipo de instrucción, estas instrucciones se pueden ver en la .

Tabla 7: Instrucciones del paquete de instrucción

Valor	Instrucciones	Descripción
0x01	Ping	Instrucción que comprueba si el paquete ha llegado a un dispositivo con la misma ID que la ID del paquete
0x02	Read	Instrucción para leer datos del dispositivo
0x03	Write	Instrucción para escribir datos en el dispositivo

0x04	Reg Write	Instrucción que registra el paquete de instrucciones a un estado de espera; El paquete se ejecuta posteriormente a través de la instrucción de Acción.
0x05	Action	Instrucción que ejecuta el paquete que se registró previamente utilizando Reg Write
0x06	Factory Reset	Instrucción que restablece la tabla de control a sus valores predeterminados de fábrica iniciales
0x08	Reboot	Instrucción que reinicia Dynamixel
0x83	Sync Write	Para múltiples dispositivos, instrucciones para escribir datos en la misma dirección con la misma longitud a la vez
0x92	Bulk Read	Para múltiples dispositivos, instrucciones para escribir datos en diferentes direcciones con diferentes longitudes a la vez

6.1.5 Parámetros

Los parámetros que se pueden utilizar para los Dynamixel están en la Tabla 8 y la Tabla 9, también podemos utilizar los registros que no se usan y escribir o leer datos de ellos.

Tabla 8: Tabla de control de parámetros para la área EEPROM

TABLA DE CONTROL PARA LA AREA EEPROM					
Dirección	Tamaño (Byte)	Nombre del dato	Descripción	Acceso	Valor inicial
0 (0x00 , 0x01)	2	Model Number	Número de modelo	R	12
2 (0x02)	1	Firmware Version	Versión del Firmware	R	-
3 (0x03)	1	ID	Identidad	R/W	1
4 (0x04)	1	Baud Rate	Velocidad de comunicación	R/W	1
5 (0x05)	1	Return Delay Time	Tiempo de retardo de respuesta	R/W	250

6 (0x06 , 0x07)	2	CW Angle Limit	Límite de ángulo a la derecha	R/W	0
8 (0x08 , 0x09)	2	CCW Angle Limit	Límite de ángulo a la izquierda	R/W	1023
11(0x0B)	1	Temperature Limit	Límite Máximo de Temperatura Interna	R/W	70
12(0x0C)	1	Min Voltage Limit	Límite de voltaje de entrada mínimo	R/W	60
13(0x0D)	1	Max Voltage Limit	Límite de voltaje de entrada máximo	R/W	140
14(0x0E , 0x0F)	2	Max Torque	Máxima Torsión	R/W	1023
16 (0x10)	1	Status Return Level	Selecciona los tipos de retorno de estado	R/W	2
17 (0x11)	1	Alarm Led	LED para alarma	R/W	36
18 (0x12)	1	Shutdown	Información de error de apagado	R/W	36

Tabla 9: Tabla de control de parámetros para la área RAM

TABLA DE CONTROL PARA LA AREA RAM					
Dirección	Tamaño (Byte)	Nombre del dato	Descripción	Acceso	Valor inicial
24 (0x18)	1	Torque Enable	Torsión del motor On/Off	R/W	0
25 (0x19)	1	LED	Estado del LED On/Off	R/W	0
26(0x1A)	1	CW Compliance Margin	Margen de error a la derecha	R/W	1

27(0x1B)	1	CCW Compliance Margin	Margen de error a la izquierda	R/W	1
28(0x1C)	1	CW Compliance Slope	Margen de error de velocidad a la derecha	R/W	32
29(0x1D)	1	CCW Compliance Slope	Margen de error de velocidad a la izquierda	R/W	32
30(0x1E , 0x1F)	2	Goal Position	Posición objetivo	R/W	-
32 (0x20 , 0x21)	2	Moving Speed	Velocidad de Movimiento	R/W	-
34 (0x23 , 0x24)	2	Torque Limit	Torque Limit(Goal Torque)	R/W	ADD 14&15
36 (0x25 , 0x26)	2	Present Position	Posición actual	R	-
38 (0x27, 0x01)	2	Present Speed	Velocidad actual	R	-
40 (0x28 , 0x29)	2	Present Load	Carga actual	R	-
42(0x2A)	1	Present Voltage	Voltaje actual	R	-
43(0x2B)	1	Present Temperature	Temperatura actual	R	-
44(0x2C)	1	Registered	Si la instrucción ha sido registrada	R	0
46(0x2E)	1	Moving	Estado del movimiento	R	0
47(0x2F)	1	Lock	Bloquear la EEPROM	R/W	0
48(0x30 , 0x31)	2	Punch	Umbral de corriente mínima	R/W	32

6.1.6 Checksum

Se utiliza para comprobar si el paquete ha sido dañado durante la comunicación. La instrucción "Checksum" se calcula de acuerdo con la siguiente fórmula.

$$Checksum = \sim(ID + Longitud + Instrucción + Parámetro1 + \ldots ParámetroN)$$

Donde "~" es el operador de Complemento de Binarios. Cuando el resultado del cálculo del paréntesis en la fórmula anterior es mayor que 255 (0xFF), se usan solo los bytes más bajos.

Por ejemplo, cuando desea utilizar el siguiente paquete de instrucciones,

ID = 1 (0x01), Longitud = 5 (0x05), Instrucción = 3 (0x03), Parámetro 1 = 12 (0x0C), Parámetro2= 100 (0x64), Parámetro3 = 170 (0xAA)

$$Checksum = \sim(ID + Longitud + Instrucción + Parámetro1 + \ldots Parámetro3)$$

$$Checksum = \sim(0x01 + 0x05 + 0x03 + 0x0C + 0x64 + 0xAA) = \sim(0x123)$$

Solo el byte inferior 0x23 ejecuta la operación de complemento a uno.

$$\sim(0x123) = 0xFEDC \rightarrow 0xDC$$

6.2 Paquete de estado o de retorno

Una vez se envía un paquete de instrucción a un motor Dynamixel u otro tipo de dispositivo, éste retorna un paquete de estado como el que aparece en la Tabla 10.

Tabla 10: Paquete de estado o de retorno general de bioloid

Encabezdo1	Encabezdo2	ID	Longitud	Error	Param 1	…	Param N	Checksum
0xFF	0xFF	ID	Longitud	Error	Param 1	…	Param N	CHKSUM

6.2.1 Error

Cuando no hay ningún problema con el envío del paquete o en el propio motor Dynamixel, el error que retornaría este paquete sería "00" , en caso contrario se puede saber cuál ha sido el fallo. Estos fallos se pueden observar en la Tabla 11.

Tabla 11: Errores posibles del paquete de estado

Bit	Error	Descripción
Bit 7	0	-
Bit 6	Error de instrucción	En el caso de enviar una instrucción indefinida o entregar la instrucción de acción sin la instrucción reg_write, se establece como 1

Bit 5	Error de sobrecarga	Cuando la carga actual no puede ser controlada por el par de torsión establecido, se establece como 1
Bit 4	Error de “checksum”	Cuando el checksum del paquete de instrucciones transmitido es incorrecto, se establece como 1
Bit 3	Error de rango	Cuando una instrucción está fuera del rango de uso, se establece como 1
Bit 2	Error de sobrecalentamiento	El valor del bit se fija en 1 cuando el valor de la temperatura sobrepasa el rango establecido.
Bit 1	Error de límite de ángulo	Cuando la posición objetivo del ángulo esta fuera de los límites, se establece en 1
Bit 0	Error de voltaje de entrada	Cuando el voltaje aplicado está fuera del rango de voltaje de funcionamiento establecido en la tabla de Control, se pone como 1

6.2.2 Checksum del paquete de estado

El checksum del paquete de estado es similar al del paquete de instrucción, quedando así la fórmula.

$$Checksum = \sim(ID + Longitud + Error + Parámetro1 + \ldots ParámetroN)$$

6.3 Ejemplos de instrucciones

A continuación se van a poner algunos ejemplos de paquetes de instrucciones y de estado, ya que dependiendo de lo que se quiera hacer hay ciertas modificaciones no especificadas.

6.3.1 Leer datos

En la Tabla 12, se muestra el paquete de instrucción para leer los datos,0x02 , de un motor Dynamixel con ID 0x01, donde “P1” es la dirección donde se guarda el valor de la temperatura y “P2” es la longitud de bytes del dato, en este caso 1.

Tabla 12: Ejemplo de paquete de instrucción de lectura

E1	E2	ID	Longitud	Instrucción	P1	P2	CKSM
0xFF	0xFF	0x01	0x04	0x02	0x2B	0x01	0xCC

La respuesta que debe devolver el motor tiene el siguiente formato, Tabla 13,donde los datos de la temperatura se encuentran en “P1”, como los datos los devuelve en hexadecimal, la temperatura del motor estaría en 32°C.

Tabla 13: Ejemplo de paquete de estado en lectura

E1	E2	ID	Longitud	Error	P1	CKSM
0xFF	0xFF	0x01	0x04	0x00	0x20	0xDB

6.3.2 Instrucción “Reg Write”

Esta instrucción es parecida a la instrucción de escribir, pero esta conlleva una sincronización con el robot posteriormente. El registro de esta instrucción se queda en espera hasta que la pueda recibir el motor, una vez pase esto, la acción se ejecuta.

Los datos que se pueden enviar puede que necesiten más de un byte, en este caso se usan los parámetros que sean necesarios del paquete de instrucción, como se ve en la Tabla 14.

Tabla 14: Ejemplo de paquete de instrucción con envío de varios bytes

Longitud	Instrucción	P1	P2	PN + 1
N + 3	0x04	Dirección inicial de los datos	1er Byte	Número N de Bytes

7 Procesado de información de la IMU

Para poder procesar la información de la IMU se tiene que entender qué tipo de datos vamos a recibir. La IMU integra varios tipos de sensores de movimiento que pueden proporcionar datos de alta precisión, éstos se pueden usar con varios fines como por ejemplo para consumidores en los teléfonos móviles, militares o industriales, para robótica. Los sensores que incorporan suelen ser los siguientes:

- Un giroscopio que mide la variación de posición angular, generalmente los valores se expresan en grados por segundo. La integración de la velocidad angular en el tiempo da como resultado la variación del ángulo de desplazamiento. Los giroscopios detectan el variación de movimiento independientemente de la gravedad por lo que los errores de integración dan como resultado un error de posicionamiento que se puede compensar con software.
- Un acelerómetro que mide la aceleración lineal donde se incluye la aceleración causada por el movimiento del dispositivo y la debida a la gravedad. La unidad de medida de este sensor es "g" donde $1\ \mathrm{g} = f.gravedad\ de\ la\ tierra = 9.8\ \mathrm{m/s^2}$. En este caso, el acelerómetro que se usa es de tres ejes, las componentes X, Y, Z.
- Un sensor magnético o magnetómetro el cual mide la intensidad del campo magnético, generalmente las unidades son micro Teslas (μT) o Gauss ($100\ \mu\mathrm{T} = 1$ Gauss). Al calcular el ángulo del campo magnético terrestre y comparándolo con el ángulo medido de la gravedad con el acelerómetro, es posible medir el rumbo respecto al norte magnético con mucha precisión.

7.1 Fuentes de error de IMU

Las principales fuentes de error las encontramos en el giroscopio y en el acelerómetro.

Los giroscopios detectan la orientación con la variación de la velocidad angular, pero suelen desviarse con el tiempo debido a que solo detectan el cambio y no tienen una referencia fija, si se incorporan los datos del acelerómetro, estos datos pueden usarse para orientar mejor el giroscopio.

Respecto al acelerómetro, son más precisos para cálculos estáticos, ya que tienden a distorsionar las aceleraciones debidas a fuerzas externas como la gravitacional. El filtrado de los datos proporciona una mejora en la precisión.

De esta forma si unimos el giroscopio, que tiene muy buena precisión a corto plazo, con el acelerómetro, que tiene gran exactitud a largo plazo, podemos compensar las debilidades de un sensor con el otro.

7.2 Filtros

Para evitar en parte las fuentes de error mencionadas se añade un filtro complementario, este combina un filtro de paso alto para el giroscopio y un filtro de paso bajo para el acelerómetro, de este modo se filtra el ruido de alta frecuencia en el acelerómetro y se suavizan los datos del giroscopio.

7.3 Tipología de los datos

Los datos de la IMU elegida se envían a través del bus I2C, se nos proporciona valores de cada eje X, Y, Z, de cada sensor, y estos datos vienen expresados en dos bytes, la parte alta del byte y la baja. El fabricante del sensor nos especifica que los datos vienen en complemento a 2.

Para aplicar el complemento a dos se debe trabajar sobre el número binario, en este se cambiarán los 1 por 0 y a viceversa (complemento a 1) y sumarle 1.

7.4 Medidas de ángulos e inclinaciones

Mediante el acelerómetro se puede obtener la inclinación de la IMU, ya que suponiendo que se ponga en una superficie plana con el eje Z podremos obtener la aceleración de la gravedad, y conforme se vaya girando la placa, la aceleración de la gravedad irá apareciendo en los otros ejes.

De esta forma podemos calcular los ángulos X e Y, pero no el Z ya que se aplica trigonometría .

Las fórmulas para calcular el resto de ángulos se muestran a continuación.

$$Ángulo Y = \text{atan}\left(\frac{x}{\sqrt{y^2 + z^2}}\right)$$

$$Ángulo X = \text{atan}\left(\frac{y}{\sqrt{x^2 + z^2}}\right)$$

Para calcular el ángulo con el cual gira la IMU mediante el giroscopio debemos guardar los datos que captura esta, compararlos con los anteriores y calcular el tiempo, de este modo si se tiene la velocidad y el tiempo durante el cual se movía la IMU, tenemos el ángulo, como se ve en la siguiente fórmula.

$$Ángulo Y = Ángulo Y Anterior + Giroscopio Y \cdot \Delta t$$

Si combinamos las medidas de los ángulos del acelerómetro y giroscopio mediante el filtro complementario, queda la siguiente fórmula.

$$Ángulo = 0.98 \times \left(Ángulo Giroscopio\right) + 0.02 \times Ángulo Acelerómetro$$

Respecto al uso del magnetómetro ,éste mide los campos magnéticos , los que pueden variar debido a elementos externos, aunque el que nos interesa es el terrestre. El campo magnético terrestre es muy débil ($25 - 65\ \mu T$) y varía con el tiempo, pero los valores obtenidos por la IMU nos permiten calcular en norte magnético con bastante precisión.

Para calcular el Norte utilizamos la siguiente fórmula:

$$\theta = \text{atan}\left(\frac{m_y}{m_x}\right)$$

Donde:

- θ: Ángulo con respecto al norte
- m_y: Valor normalizado del magnetómetro en el eje y
- m_x: Valor normalizado del magnetómetro en el eje x

Este ángulo nos mide el norte magnético, pero existe una diferencia con el norte geográfico, a esta diferencia se le llama como declinación magnética, Figura 56.

El valor de la declinación magnética depende de la posición en la que estés, pero como no se necesita situar el norte geográfico para la orientación de nuestro robot, la fórmula anterior nos sirve.

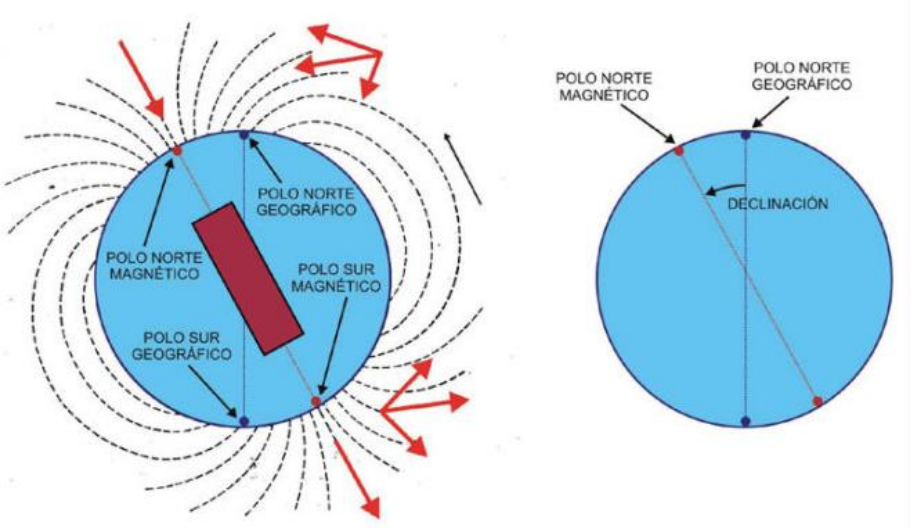


Figura 56: Declinación magnética

Mediante los tres sensores se nos permite calcular la posición en el espacio del robot y además tener una referencia de posición inicial usando el magnetómetro.

8 Sensado de la batería

Al introducir un sensor de intensidad y un divisor de tensión, se nos permite saber el nivel de la batería y la cantidad de energía que se ha extraído de ésta, a continuación se especifican estos métodos.

8.1 Nivel de batería

El nivel de tensión en el caso de las baterías de litio, como es el caso, baja de forma lineal conforme va disminuyendo la carga de la batería, esto nos permite poder saber el nivel de la batería de forma sencilla mediante un divisor de tensión. El divisor de tensión nos permite saber que tensión hay en la batería con respecto a la fórmula que se muestra a continuación:

$$V_S = V_e \times \frac{R1}{R2 + R1}$$

Donde:

- V_S : Tensión que queremos en la salida
- V_e : Tensión en la batería
- $R1$: Resistencia en la salida para ajustar la tensión en ésta junto a $R2$
- $R2$: Resistencia de ajuste de tensión

La tensión en la entrada será la de la batería, que en carga máxima es de 12 V, y la salida tensión en la salida como máximo será 3.3 V, debido a que el microprocesador funciona a 3.3 V.

Mediante la tensión de entrada y salida se calculan las resistencias, los resultados de las cuales se pueden observar en la .

8.2 Estado de la batería

Para conocer el estado de la batería se usa el sensor de intensidad de efecto hall que nos proporciona una salida en tensión con respecto a la corriente que transcurre por el sensor.

Según la hoja de datos del sensor, éste funciona con corrientes positivas y negativas, aunque a nosotros solo se nos proporcionaran corrientes positivas, para ello se tiene un nivel de tensión inicial cuando la corriente es 0 igual a $V_{cc}/2$ o si la adaptamos a nuestro microprocesador $V_0 = 3.3 \,/\, 2 \;=\; 1.65$ V.

A partir del nivel inicial de la batería se va aumentando la tensión conforme aparece en las hojas de datos, en éstas se nos indica que por cada 110 mV en la salida, aumenta 1 A en la entrada.

Para conocer si el estado de la batería es correcto comprobamos la capacidad de la batería que usaremos, esta es de 1000 mAh, esto nos indica que la batería es capaz de proporcionar 1000 mA en 1 hora, si no se proporcionara esta capacidad sabremos que la batería se ha deteriorado. Para realizar la medida tenemos que referir la corriente que pasa a través del robot con el tiempo que ha estado funcionando la batería, para ello utilizamos las funciones de medida de tiempo que se han creado en el microprocesador y una vez se haya agotado la batería se puede conocer el estado de la batería.

9 Software

Para realizar el software del microprocesador, se ha utilizado el programa MPLAB X IDE, que es un software de los fabricantes del microprocesador (Microchip), este se basa en C++.

En primer paso se ha procedido a realizar la inicialización del pic32 mediante un archivo denominado “config.h”, el propio software es el que nos genera este archivo que depende de los parámetros seleccionados y del dispositivo usado.

A continuación se inician varias librerías creadas a propósito para esta PCB, entre las que se incluyen:

- Los ledes

- Los pulsadores, que funciona mediante interrupciones asignadas a los pines de esto en flanco negativo

- El módulo del oscilador interno para poder realizar mediciones de tiempo, este módulo funciona mediante interrupciones cada milisegundo

-La comunicación UART, esta librería también funciona mediante interrupciones para la recepción, debido a que es un protocolo muy rápido.

-La comunicación I2C

Una vez realizadas las inicializaciones de todas las librerías, se procede con el programa principal, que se puede resumir en la Figura 57.

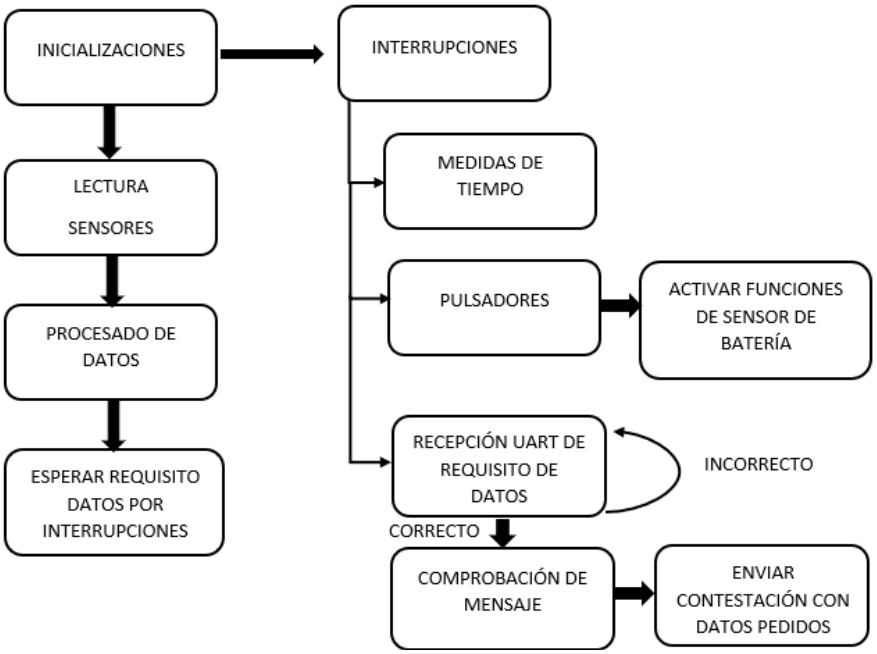


Figura 57: Esquema del programa de la PCB

Respecto a la programación del robot, se utiliza el software del fabricante del robot, “Roboplus task” , que se basa en una programación C asistida. Para poder demandar los datos de la PCB como nos comunicamos mediante el bus de motores, tenemos que demandar los datos como si se tratara de otro motor, para ello asignamos una ID ficticia a nuestros datos y los situamos en registros de esa ID, en la Figura 58 se puede observar cómo se puede hacer esto mediante el software del robot, donde la ID ficticia sería la 50 y la dirección del registro la 70.

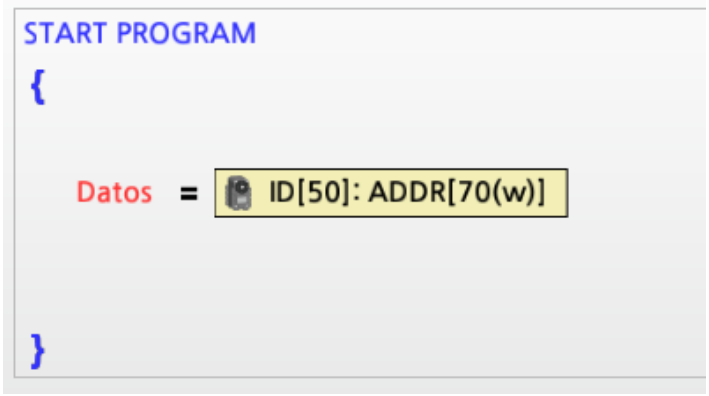


Figura 58: Ejemplo de requisito de datos del robot

Mediante el uso de los datos, se realizan los movimientos necesarios para orientarse, los cuales se hacen mediante otro programa del fabricante, “Roboplus motion”, en este programa se programan los movimientos que se quieran realizar para luego ser “llamados” mediante el anterior programa.

10 Resultado obtenido

Para poder montar la PCB sobre el robot bioloid, primero se debe saber cómo se conectará la PCB al robot bioloid. Mediante la batería que está situada en la espalda del robot, Figura 59, se conectará a la PCB para que el sensor de intensidad mida la corriente total que necesita el robot, a partir de ahí se alimenta la CPU del robot bioloid, la cual tiene varios conectores que tienen una alimentación de 12 V (Pin 2), una masa o GND (Pin 1) y un bus de datos a 5 V (Pin 3), Figura 60.

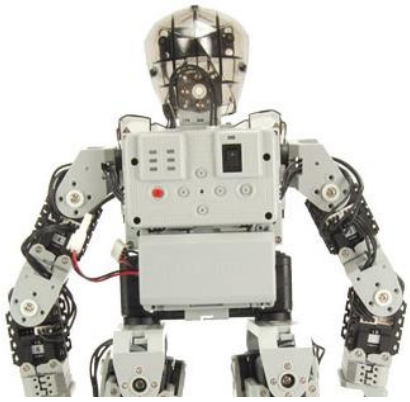


Figura 59: Parte trasera del robot bioloid

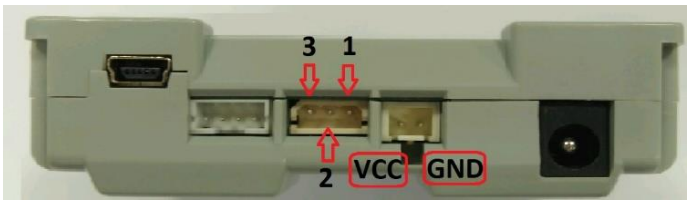


Figura 60: Conectores CPU del robot bioloid

A partir del conector mencionado alimentamos a la PCB, que estará situada en el pecho del robot, esta parte esta desprotegida, para ello se ha diseñado una caja de protección, que a la vez nos permite sujetarla al robot, Figura 61.

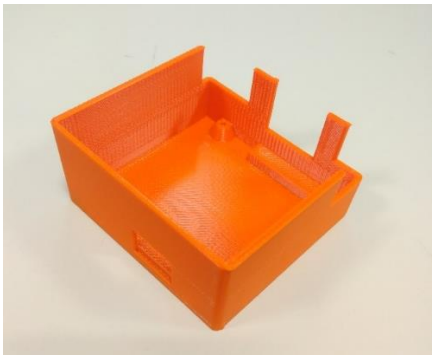


Figura 61: Caja de protección de la PCB

Finalmente en la Figura 62, aparece el resultado final.

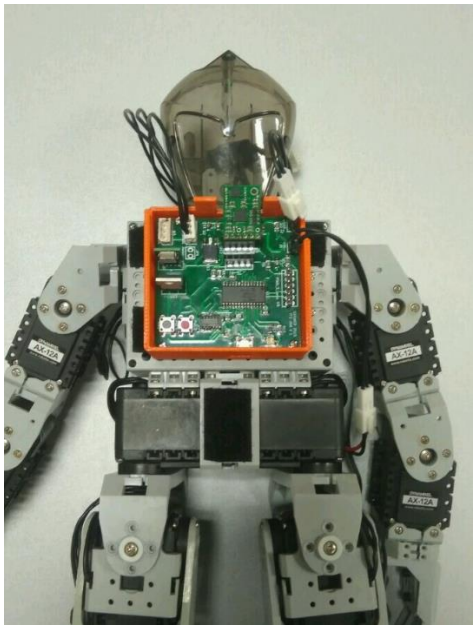
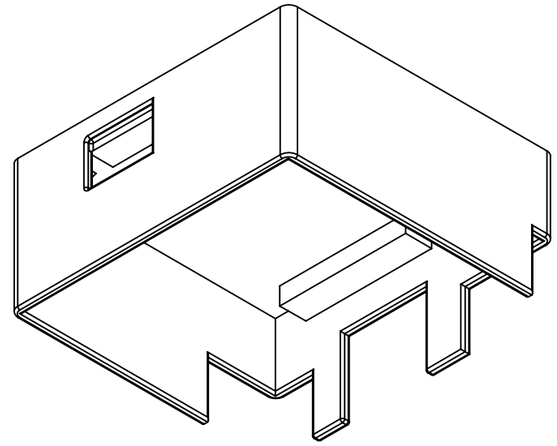
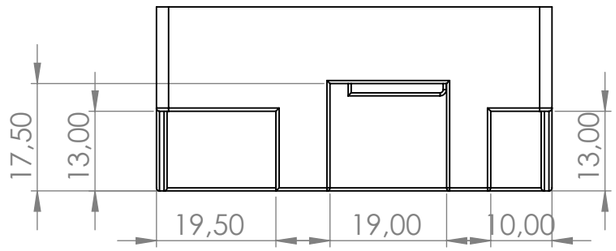
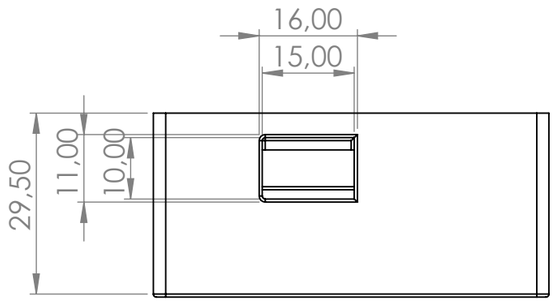


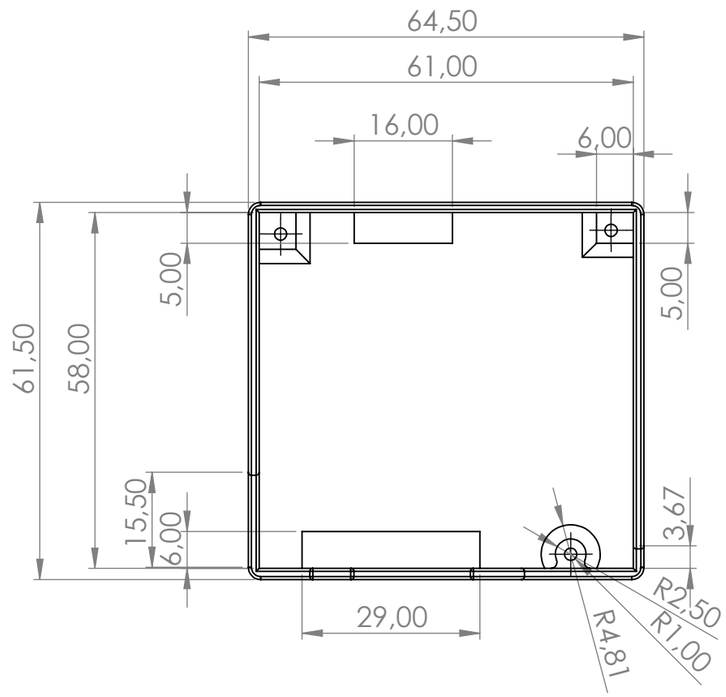
Figura 62: Resultado final del montaje de la PCB en el robot bioloid

11 Planos

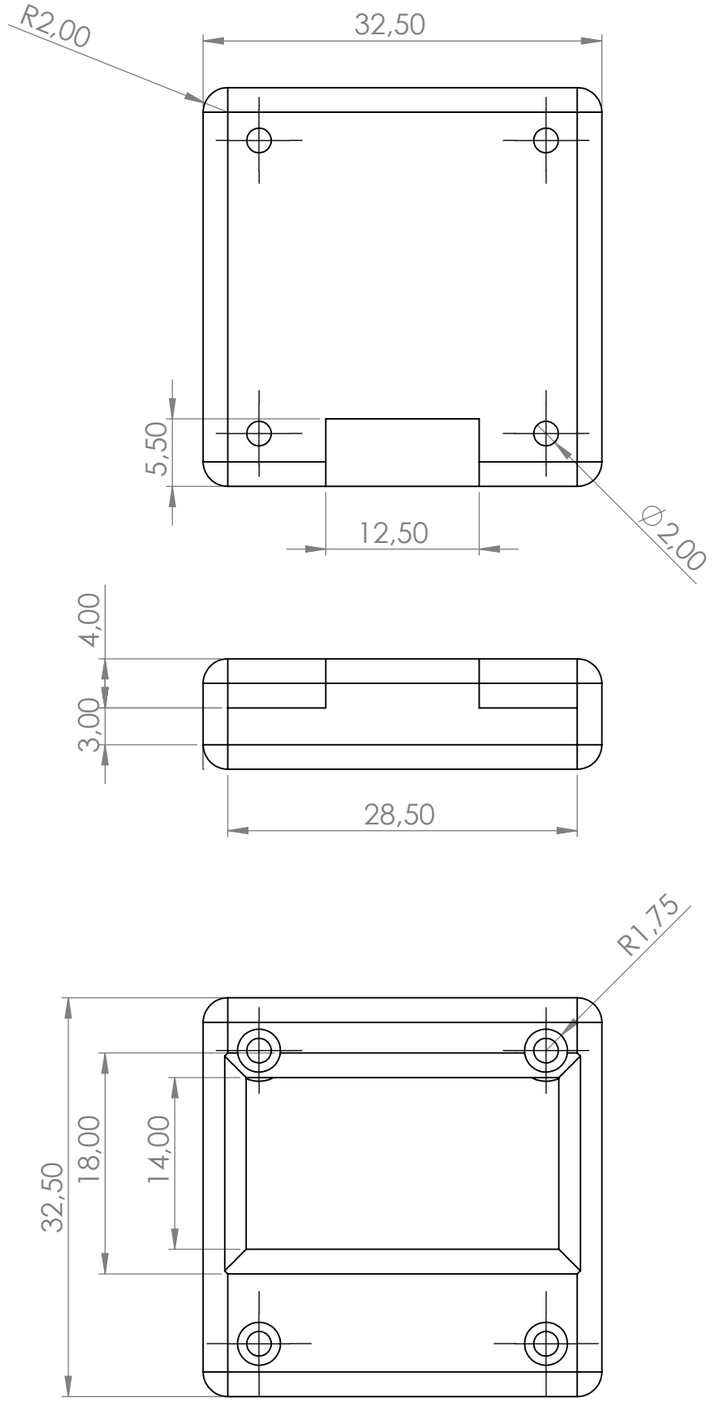


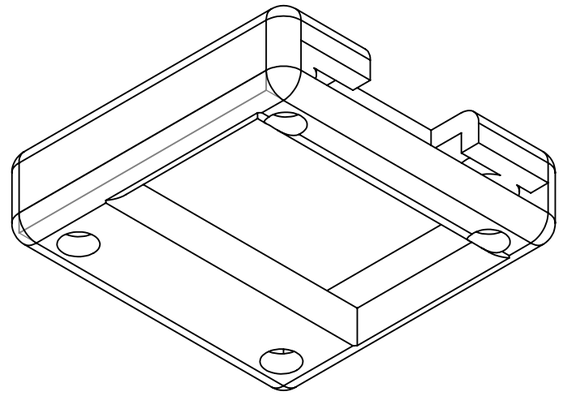






	EL AUTOR DEL PROYECTO Jordi Armengol Miralles	FIRMA
UNIVERSITAT POLITÈCNICA DE VALÈNCIA CAMPUS ALCOI		FECHA
		SUSTITUYE A
TITULO	Diseño de pcb para orientación de un robot y sensado de la batería	SUSTITUIDO POR
SITUACION		
ESCALAS 1:1	TITULO DEL PLANO Caja Protectora PCB	Nº PLANO 1





	EL AUTOR DEL PROYECTO: Jordi Armengol Miralles	FIRMA
UNIVERSITAT POLITÈCNICA DE VALÈNCIA CAMPUS ALCOI		FECHA
		SUSTITUYE A
TITULO: Diseño de pcb para orientación de un robot y sensado de la batería		SUSTITUIDO POR
SITUACION		
ESCALAS: 2:1	TITULO DEL PLANO: Carcasa Display	Nº PLANO: 2

12 Conclusiones

Respecto al diseño de la PCB, se ha conseguido que la placa de circuito impreso sea funcional para la conexión entre los componentes y el microprocesador, así mismo nos a permitido conectarnos con los sensores de orientación mediante el protocolo de comunicaciones I2C y con el robot bioloid mediante la comunicación serie.

Una vez realizadas las conexiones se ha conseguido extraer los datos de la IMU, para esto se han configurado los protocolos de comunicación en el microprocesador y a continuación mediante el formato especificado por el fabricante se ha obtenido la información de orientación.

Los datos obtenidos por los sensores se han procesado para utilizarlos de forma sencilla y así que el robot los pueda usar.

Para enviar los datos al robot bioloid se ha utilizado el bus de datos serie que usa el propio dispositivo para conectar sus motores y mediante el formato específico de envío y recepción de datos de los motores se han comunicado los datos procesados de la IMU.

A través del sensor de intensidad colocado en la placa de circuito impreso y el divisor de tensión se ha podido medir el nivel de tensión de la batería para conocer la carga restante y conocer el estado de la batería referente a la pérdida de capacidad que ocurre con el paso del tiempo.

Finalmente se ha conseguido integrar la PCB con el robot bioloid y que mediante los datos proporcionados por los sensores de la IMU se oriente en el espacio.

13 Bibliografía

Motores Dynamixel de robot bioloid:

http://emanual.robotis.com/docs/en/dxl/ax/ax-12a/

Robot bioloid premium.

http://emanual.robotis.com/docs/en/edu/bioloid/premium/

Protocolo de comunicaciones Dynamixel:

http://emanual.robotis.com/docs/en/dxl/protocol1/

CPU robot bioloid:

http://support.robotis.com/en/product/controller/cm530.htm

Información sobre la IMU:

https://www.digikey.es/es/articles/techzone/2019/apr/imus-let-your-host-sleep-with-on-board-machine-learning

Manual del microprocesador:

http://ww1.microchip.com/downloads/en/DeviceDoc/PIC32MX1XX2XX-28-36-44-PIN-DS60001168K.pdf

Uso del magnetómetro:

http://blascarr.com/lessons/introduccion-al-imu-sistemas-de-navegacion-inercial/

Bruce Ekel:" **Thinking in C++"**, 1995. ISBN: 978-0139177095