UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO DE UNA INTERFAZ HOMBRE MÁQUINA BASADA EN JOYSTIC PARA EL CONTROL DE APLICACIONES ANDROID

Ingeniería Electrónica Industrial y Automática

Autor: Manuel Benedicto García

Tutor: Carlos Pascual Domínguez

Curso académico: 2017-2018

AGRADECIMIENTOS.

A mi tutor, el profesor Carlos Domínguez Montadug, por su gran ayuda y colaboración en los momentos de dudas, y por el apoyo mostrado a este proyecto.

A los profesores Pedro Ramón Podadera Sánchez y Guillermina Tormo Carbó, por tratarse de dos profesores excepcionales, capaces de sacar lo mejor de mí, y convertirme en un mejor ingeniero.

A mi compañero de grado y de fatigas Ruslan Bondar, por la ayuda prestada durante el transcurso del grado y de la vida.

A mi pareja, Miriam Romera García, por ser un apoyo constante, dándome la confianza y la ayuda necesaria para llegar hasta aquí.

Y por último, a mi padre, Manuel Benedicto Soler, por realizar incontables esfuerzos por mi y por mi futuro, permitiendo que la escritura de estas líneas haya sido posible.

Resumen– Castellano

El proyecto consiste en el diseño de un mando de control remoto, que está formado por un joystick y un pulsador, para controlar aplicaciones en un terminal Android.

En primer lugar, se ha diseñado la parte electrónica, montando el circuito del microprocesador Arduino Uno con el módulo de comunicación bluetooth HC-06, el joystick, de la marca Robotlinking, y un pulsador convencional.

En segundo lugar, se ha programado el código de Arduino de manera que detecte la variación de señal del joystick mediante una señal analógica y la transforme en una señal digital que vaya de 0-99 y envíe esta señal mediante el módulo bluetooth.

Por último, mediante el entorno de desarrollo de aplicaciones para dispositivos Android MIT App Inventor 2, se ha diseñado una aplicación que lea la señal enviada por el Arduino Uno y la interprete, interactuando con el usuario mediante un videojuego.

Palabras clave: Joystick, Arduino, Bluetooth, Android.

Resum – Valencià

El projecte consisteix en el disseny d'un comandament de control remot, que està format per un joystick i un polsador, per controlar aplicacions en un terminal Android.

En primer lloc, s'ha dissenyat la part electrònica, muntant el circuit del microprocessador Arduino U amb el mòdul de comunicació bluetooth HC-06, el joystick, de la marca Robotlinking, i un polsador convencional.

En segon lloc, s'ha programat el codi d'Arduino de manera que detecte la variació de senyal del joystick mitjançant un senyal analògic i la transforme en un senyal digital que vaja de 0-99 i envie aquest senyal mitjançant el mòdul bluetooth.

Finalment, mitjançant l'entorn de desenvolupament d'aplicacions per a dispositius Android MIT App Inventor 2, s'ha dissenyat una aplicació que llegisca el senyal enviat per l'Arduino U i l'interprete, interactuant amb l'usuari mitjançant un videojoc.

Paraules clau: Joystick, Arduino, Bluetooth, Android.

Abstract

The project consists of the design of a remote control, which is formed of a joystick and a button, to control applications on an Android terminal.

First of all, the electronic part has been designed assembling the circuit of the Arduino Uno microprocessor with the bluetooth communication module HC-06, the joystick, of the Robotlinking brand, and a conventional pushbutton.

Secondly, the Arduino code has been programmed to detect the signal variation of the joystick by an analog signal and transforms it into a digital signal that goes from 0-99 values and sends this signal through the bluetooth module.

Finally, through the application development environment for Android devices MIT App Inventor 2, has been designed an application that reads the signal sent by the Arduino Uno and interprets it, interacting with the user by a videogame.

Keywords: Joystick, Arduino, Bluetooth, Android.

Índice de documento

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO DE UNA INTERFAZ HOMBRE MÁQUINA BASADA EN JOYSTIC PARA EL CONTROL DE APLICACIONES ANDROID

Trabajo Fin de Grado en Ingeniería Electrónica Industrial y Automática

Documento 1: Memoria

Autor: Manuel Benedicto García

Tutor: Carlos Pascual Domínguez Montagud

Índice general

Contenido

Índice de tablas

Índice de figuras

1.Introducción

1.1. Motivación del proyecto

Con el creciente avance tecnológico que hay hoy en día, es difícil encontrar a alguien que no posea un teléfono móvil, y esto es algo que las empresas de los videojuegos han empezado a explotar en los últimos años. Si bien utilizan un modelo de mercado muy distinto al usado por el mercado de las videoconsolas, consiguen obtener la misma cantidad de beneficio e incluso en algunos casos destacados mayor beneficio. Mientras que en cualquier videojuego de videoconsola el consumidor debe comprar el videojuego pagando una cantidad monetaria importante, puede oscilar entre 15 y 75 euros, los videojuegos de Smartphone suelen ser gratuitos, los denominados "free to play", en los que el consumidor no se ve obligado a comprar nada, pero sí contiene pequeñas compras en el juego que ayudan al consumidor a avanzar en el videojuego de manera más rápida, dichas compras son las denominadas "micropagos", y suelen ser de un coste relativamente bajo, oscilando entre 1 y 10 euros, de manera que la sensación de desembolso en el cliente sea más baja, y por lo tanto sea más propenso a comprar.

Si bien el beneficio es mucho menor, la clave de este tipo de ventas es la gran cantidad de potenciales clientes que tienen los juegos de Smartphone, ya que, mientras un videojuego de videoconsola sólo será comprado por consumidores que ya posean dicha videoconsola, los artículos de los videojuegos de Smartphone pueden ser adquiridos por prácticamente toda la población, lo cual lleva a una gran cantidad de consumidores realizando microcompras, y por lo tanto, unos grandes beneficios. En la siguiente imagen se pueden apreciar algunos de los casos más destacados de ganancias con microventas de artículos en videojuegos de Smartphone durante el primer mes de vida de dichos videojuegos.

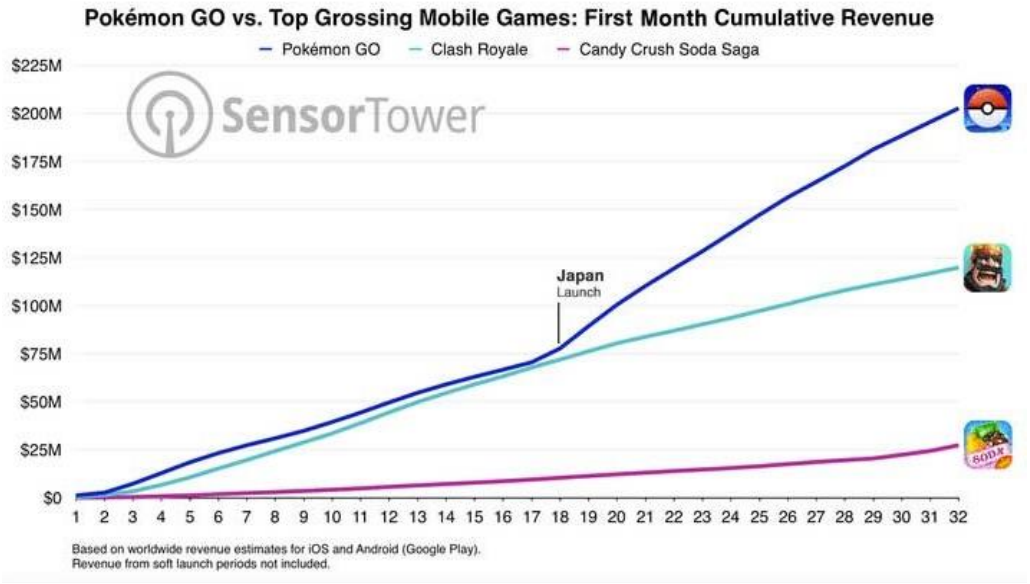


Figura 1. Ganancia con micro pagos de varios juegos durante su primer mes de lanzamiento

Como se ha podido observar, el mercado de los videojuegos tiene una gran cantidad de clientes potenciales, pero también tiene un gran inconveniente, y es que, las características de cada Smartphone varían según el fabricante y el modelo (por ejemplo, el tamaño de la pantalla, la sensibilidad de la pantalla, la velocidad de procesamiento, etc.) dificultando así el diseño de los videojuegos y también la jugabilidad del cliente, ya que, un jugador puede acostumbrarse a jugar con un modelo específico de Smartphone por sus características, y si después tiene que cambiar de teléfono móvil por cualquier circunstancia deberá de aclimatarse al nuevo dispositivo. Por todos estos motivos se ha tratado de unificar los controles de jugabilidad de todos los teléfonos, diseñando un mando de control de smarthpone que permita, por un lado, poder diseñar los videojuegos sin las grandes complicaciones de tener que hacerlo compatible para todo tipo de controles, y por otro lado, ayudar a los jugadores a tener una mayor sensación de comodidad al jugar (resulta más cómodo controlar un mando cuya forma ergonómica está diseñada para ser utilizado exclusivamente en videojuegos, que no un teléfono, cuya forma está diseñada para muchas otras actividades) a la vez que una mayor adaptabilidad a cualquier tipo de teléfono móvil, ya que lo único con lo que tendrán que controlar el Smartphone será con un mando, el cual será universal para cualquier versión de móvil, sin importar el proveedor ni el modelo.

2. Contexto Tecnológico

2.1 Microprocesadores

Para hablar del contexto tecnológico de los microprocesadores primero hay que conocer qué es exactamente un microprocesador, el cual se define como la unidad base y central del procesamiento de datos, constituido solamente por un circuito integrado. El microprocesador recibe las instrucciones mediante un código de programación y actúa en consecuencia.

En varias fuentes se afirma que el lanzamiento del primer microprocesador integrado se sitúa en el 15 de noviembre de 1971, cuando la empresa Intel, que por aquel entonces todavía era una compañía pequeña, diseñó el microprocesador Intel 4004 por encargo de un fabricante japonés de calculadoras llamado Busicom. Pero lo cierto es que el primer microprocesador integrado, acorde a la definición mencionada en el párrafo anterior, fue diseñado por Gary Boone y Michael Cochran, dos ingenieros de Texas Instruments, quienes, tratando de reducir al máximo el coste de sus calculadoras electrónicas, y al mismo tiempo aumentar su fiabilidad, juntaron en un solo chip todos los circuitos digitales que componían dichas calculadoras. Texas Instruments patentó el microprocesador en un único circuito integrado el 31 de agosto de 1971, según aparece en el registro americano U.S. Patent 3.757.306, y comenzó su comercialización el 17 de septiembre de 1971, siendo el primer microprocesador el TMS1802NC.

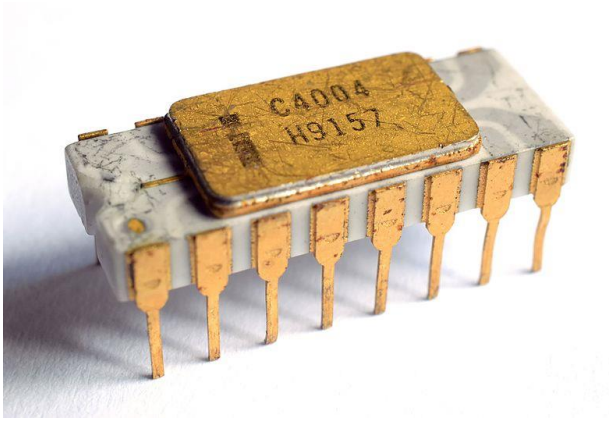


Figura 2. Procesador Intel C4004

Figura 3. Microcontrolador

Así fue como se inició la carrera tecnológica de los últimos años con la aparición de los sistemas electrónicos programables. Para poder hacerse una idea de las características de aquellos procesadores en comparación con los de hoy en día se han recogido en la Tabla 1 las características de varios microprocesadores de distintas épocas.

Microprocesador	Intel 4004	Intel 80286	DEC Alpha	Intel Core i7
Año Lanzamiento	1971	1980	1992	Actualidad
Nº Transistores	2300	134000	1.680.000	>700.000.000
Frecuencia de reloj	700 kHz	25 MHz	200 MHz	3 GHz
Arquitectura	4 bits	16 bits	64 bits	64 bits

Tabla 1. Evolución de los microprocesadores.

Para poder comprender el gran avance que supuso dicha innovación en la tecnología del momento basta con entender que, a partir de este momento, pasaron a existir unos dispositivos que permitían cambiar su función mediante un cambio en el código de programación, sin necesidad de cambiar el dispositivo en sí, evitando así un gasto en componentes y permitiendo así la posibilidad de probar infinitas configuraciones de código distintas hasta dar con la más adecuada para cada caso.

Tras la aparición de los microprocesadores, llegaron los microcontroladores. Los microcontroladores constan de distintos bloques, donde cada cual cumple una tarea específica. Los tres bloques principales son:

- La unidad central de procesamiento, o lo que es lo mismo, el microprocesador.
- La memoria, formada por un módulo de memoria RAM, que permite almacenar datos a corto plazo, los cuales son borrados cuando se deja de alimentar al microcontrolador, y por un módulo de memoria ROM, para almacenar los datos a largo plazo.
- Los periféricos de entrada/salida, los cuáles permiten la comunicación con el exterior del sistema.

El primer microcontrolador que apareció en el mercado fue el TMS1000 de Texas Instruments, el cual fue una derivación del microprocesador TMS1802NC visto anteriormente. De igual manera a la que se ha podido observar en la Tabla 1, el avance en el desarrollo de los microcontroladores también fue exponencial, haciéndolos cada vez más pequeños y con un coste más bajo, y mejorando sus prestaciones tanto en velocidad de procesado como en capacidad de memoria.

El abaratamiento de dicha tecnología fue acercando cada vez más y más la electrónica a los usuarios de nivel medio, llegando a su punto álgido con la aparición de las primeras placas de desarrollo, las cuáles incluían un microcontrolador con infinidad de posibilidades, debido a que tenían multitud de periféricos distintos, como por ejemplo sensores de temperatura, sensores de humedad, sensores de proximidad, etc.

La gran ventaja que tenían las placas de desarrollo era que eliminaban la problemática que los componentes electrónicos podían ocasionar a un usuario de nivel medio, como podrían ser el ruido o los malos acoplamientos. La sencillez en el uso de dichas placas de desarrollo hizo que tuvieran una gran acogida y fueran avanzando hasta hoy en día, dónde el más claro exponente de dichas placas es el famoso Arduino.

Arduino es una plataforma de hardware libre con un microcontrolador Atmel AVR, montado en una PCB con los componentes básicos para su correcto funcionamiento, diseñada para proyectos multidisciplinares. Además, dispone de un entorno de desarrollo libre y un lenguaje de programación propio y simple, lo cual dio la opción de acercar la electrónica a personas de nivel medio con escaso conocimiento sobre microcontroladores.

El proyecto Arduino se retoma al año 2005, cuando unos estudiantes del Instituto de diseño Interactivo IVREA, en Italia, mientras trabajaban con el microcontrolador BASIC Stamp, basado en PIC, con un lenguaje de programación simplificado, pero con un coste muy elevado, decidieron trabajar en el desarrollo de un sistema Open Hardware para abaratar el coste de este tipo de productos para hacerlo llegar a usuarios de poder adquisitivo medio-bajo.

Figura 4 Logo de Arduino

Hernando Barragán, estudiante colombiano, desarrolló la tarjeta electrónica conocida como Wiring, el lenguaje de programación, y la plataforma de desarrollo. Basado en ese trabajo, Massimo, Davir Cuartilles, y Gianluca Martino desarrollaron una plataforma de hardware y software libre, más pequeña y más económica, a la que llamaron Arduino.

El proyecto siguió avanzando y fueron apareciendo más y más modelos hasta que en la feria Maker Fair de 2011 se presentó la primera placa Arduino de 32 bits para trabajar tareas más pesadas.

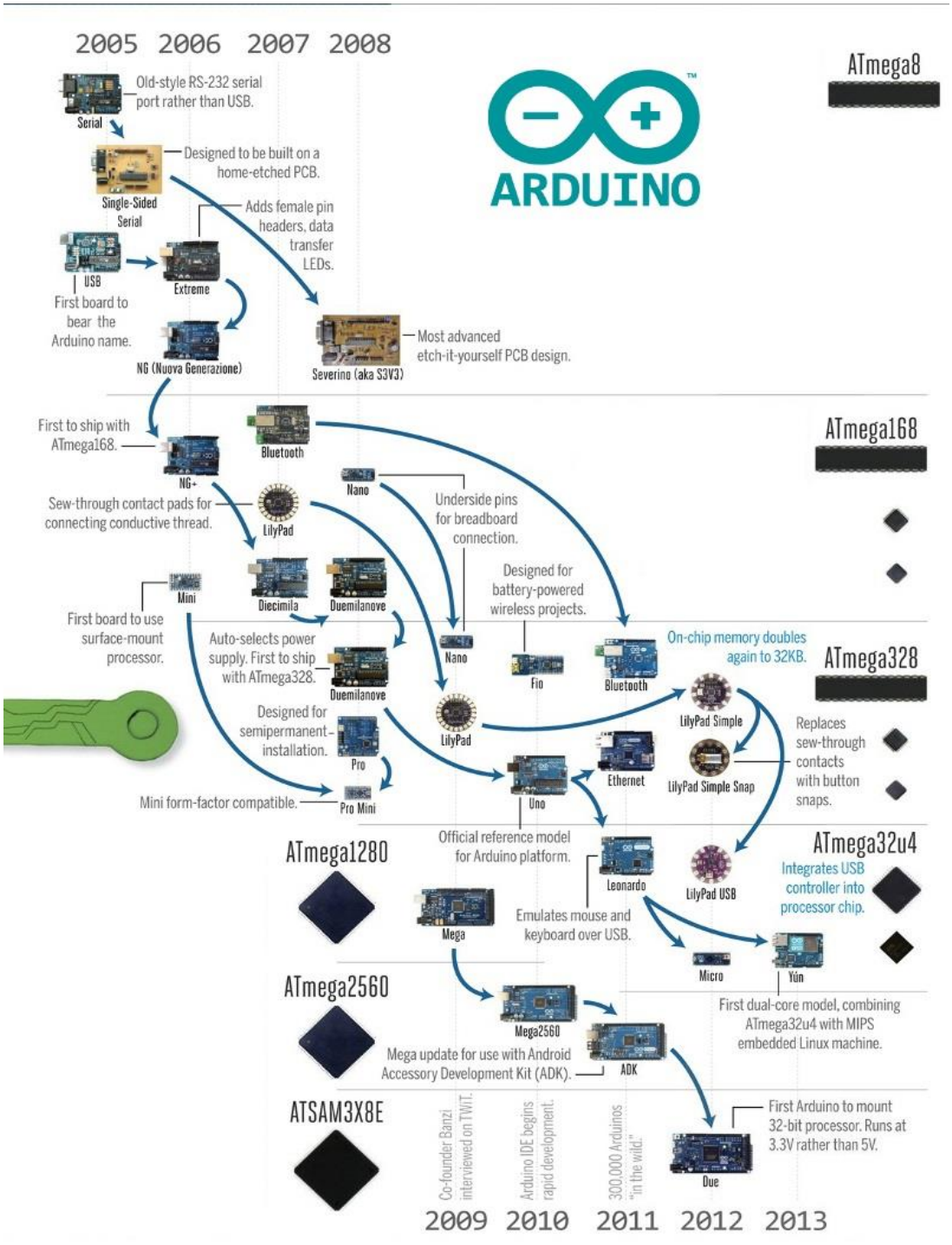


Figura 5. Evolución de Arduino

En el presente proyecto se trabajará con el microcontrolador Arduino Uno, cuyas características se pueden apreciar en la Tabla 2.

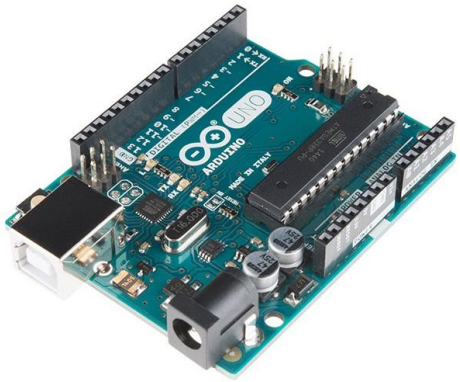


Figura 6. Arduino Uno

Microcontrolador	Atmega328
Voltaje de operación	5V
Voltaje de entrada (Recomendado)	7 – 12V
Voltaje de entrada (Límite)	6 – 20V
Pines para entrada- salida digital.	14 (6 pueden usarse como salida de PWM)
Pines de entrada analógica.	6
Corriente continua por pin IO	40 mA
Corriente continua en el pin 3.3V	50 mA
Memoria Flash	32 KB (0,5 KB ocupados por el bootloader)
SRAM	2 KB
EEPROM	1 KB
Frecuencia de reloj	16 MHz

Tabla 2. Características técnicas de Arduino Uno.

2.2 Comunicación inalámbrica

La invención del telégrafo en 1847 y el teléfono, unos años más tarde en 1876, son los dos sistemas que inician lo que se denomina comunicaciones a larga distancia. Pero existía un gran inconveniente, que era la limitada infraestructura de cableado, por lo que algunas zonas geográficamente inaccesibles permanecían incomunicadas.

Heinrich Rudolf Hertz, un físico alemán nacido en Hamburgo y que estudió en la Universidad de Berlín fue quien, en 1887, demostró que la electricidad podía ser transmitida por medio de ondas electromagnéticas a través del aire, hoy conocidas como ondas hertzianas. Dicho experimento, que necesitó de un oscilador y un resonador, confirmó las ideas de Maxwell y mostró la posibilidad de producir y transmitir ondas eléctricas a distancia y recuperarlas mediante un aparato receptor.

No hubo grandes avances hasta 1890, cuando el físico francés Edouard Désiré Branly diseñó un dispositivo llamado cohesor, que permitía comprobar la presencia de ondas radiadas, o lo que es lo mismo, desarrolló un sistema detector de señales electromagnéticas.

Con el detector de Brandly, y el oscilador de Herzt trabajando como emisor, ya sólo faltaba un receptor para poder tener todas las partes de un sistema de comunicación. Y así fue, como en 1895, el ingeniero ruso Aleksandr Stepanovich Popov, inventó la antena radioeléctrica, construyendo así el primer receptor, y logrando con ello establecer las primeras transmisiones inalámbricas a una larga distancia. Un tiempo más tarde, el 24 de marzo de 1896, Popov transmitió el primer mensaje telegráfico entre dos construcciones de la Universidad de San Petersburgo, situadas a una distancia de 250 metros. El texto de este primer mensaje telegráfico inalámbrico fue simple: "Heinrich Hertz".

Así pues, con los tres elementos ya diseñados, solo faltaba alguien capaz de juntarlos para diseñar un sistema de comunicación inalámbrico, y la persona que lo logró fue el físico italiano Guillermo Marconi, quien realizó experimentos exitosos de comunicación inalámbrica y consiguió la primera patente en la Gran Bretaña el 2 de julio de 1897, por lo cual se le considera el padre de la radio y de las telecomunicaciones inalámbricas.

La radio fue la madre de las comunicaciones inalámbricas, y a partir de su invención en 1901 los avances en este terreno han sido abrumadores, llegando a las comunicaciones inalámbricas que hay hoy en día como pueden ser los infrarrojos, el WiFi, o el Bluetooth.

En este último se va a centrar más la introducción a partir de ahora, debido a que se ha trabajado con él en el proyecto.

El nacimiento de Bluetooth se sitúa en la década de los 90', cuando Ericsson se trató de desarrollar una tecnología capaz de realizar comunicaciones a distancias cortas con la ventaja de emplear una cantidad de energía bastante pequeña en los dispositivos (principalmente móviles). Ese proyecto fue bautizado como MCLink.

Varios de los gigantes del sector, como Apple, Ericsson, Intel, Lenovo, Microsoft, Motorola, Nokia, Nordic Semiconductor y Toshiba, formaron un SIG (Special Interest Group), lo que vendría a ser una especie de grupo dónde las empresas colaboradoras aportan capital monetario y humano para favorecer el avance de una tecnología de la que después harán uso.

El nombre de Bluetooth hace honor al rey danés Herald Blåtand, cuya traducción de su apellido a inglés sería “Bluetooth”, personaje de notable relevancia escandinava en la época feudal. Además, el logo es una combinación de dos letras del alfabeto rúnico. Precisamente la H y la B.

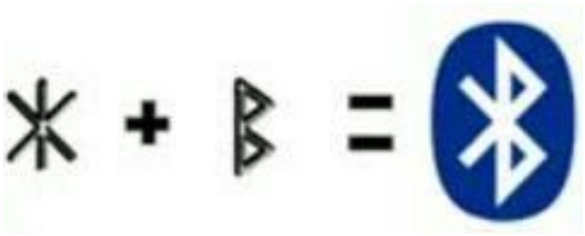


Figura 7. Origen del logo de Bluetooth

Así pues, Bluetooth fue una realidad en 1998, y por aquel entonces mucha gente pensó que Bluetooth iba a ser una competencia para WiFi, pero nada más lejos de la realidad, Bluetooth tenía unos objetivos finales muy marcados por el SIG, que eran establecer conexiones que requiriesen poco gasto de energía, por norma de corta duración, otorgando una seguridad mediante claves de cifrado y contraseña para establecer conexión, y, además, tener un bajo coste.

Finalmente, la primera versión de Bluetooth fue lanzada en 1999, la cuál permitía una velocidad teórica de hasta 0.8-1Mbps a una distancia menor de 10 metros, aunque la realidad era otra, alcanzando por norma general 720 kbps. Nada que ver con la versión más reciente que llegó al mercado en 2016, que es la versión Bluetooth 5.0 y que ha llegado a alcanzar velocidades de hasta 24 Mbs y es capaz de trabajar incluso a 240 metros de distancia.

2.3. Software

A día de hoy, con el surgimiento de Android, se puede asegurar que Java es uno de los lenguajes con uno de los usos más extendidos mundialmente, estableciéndose como el lenguaje de programación de móviles más utilizados del planeta.

Pero para hablar de la historia de Java hay que retroceder hasta la década de los años 80', en los cuáles C era en lenguaje de programación predominando, debido a que era un lenguaje versátil, capaz de actual a bajo nivel y resolver problemas complejos, pero para alcanzar la solución de estos algoritmos eran necesarios grandes procedimientos con un código complicado de mantener a largo plazo. Así pues, surgió una programación orientada a objetos como alternativa a la programación en C, llamada C++.

En ese contexto nace Java, en 1991, con el nombre de "OAK", que después sería sustituido por problemas legales por el nombre de Java.

La definición de más correcta de Java se ha encontrado en Wikipedia, y dice así, "Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra."

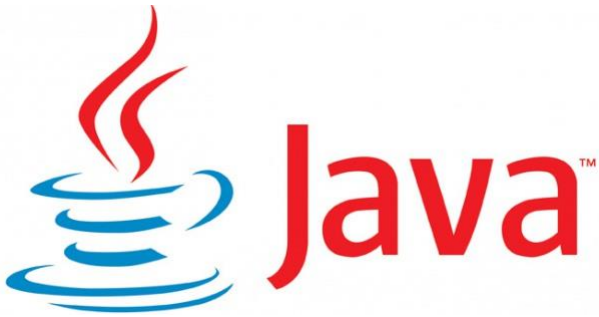


Figura 8. Logotipo de Java

En 1995 fue presentada la versión alpha de Java, y tan sólo un año después fue lanzado el primer JDK (JDK 1.0). Desde entonces hasta el día de hoy el desarrollo de Java ha sido imparable apareciendo nuevos paquetes y librerías constantemente.

Para ayudar a simplificar aún más la programación en Java, y con ello, el desarrollo de aplicaciones funcionales con el sistema operativo Android, Google Labs diseñó un entorno de desarrollo de software en el cual un usuario medio puede encajar una serie de bloques de forma visual y mediante un conjunto de herramientas básico, creando así la aplicación. En sistema, que salió al público el 25 de diciembre de 2008, fue bautizado como App Inventor, siendo totalmente gratuito, y a pesar de que las aplicaciones desarrolladas en este entorno están ciertamente limitadas por su simplicidad, son suficientes para cubrir una gran cantidad de necesidades básicas en un dispositivo móvil.

Figura 9. Logotipo de MIT APP Inventor

3. Análisis y desarrollo del problema

3.1 Análisis de los requisitos y propuesta de posible solución.

Los requisitos del prototipo consisten en obtener un mando de control remoto sobre un teléfono móvil, y para ello primero se tienen que analizar los sistemas que integran el proyecto, que incluye un sistema de control, un sistema de comunicación, y un sistema de recepción de datos.

Dentro de cada sistema hay varios requisitos distintos, por ejemplo, en el sistema de control un requisito indispensable será que sea capaz de controlar a una velocidad adecuada para la vista humana. La cantidad de imágenes que puede captar el ojo humano se expresa en fotogramas por segundo (en inglés frames per second y abreviado FPS) o en hercios (Hz). La cantidad de fps que puede llegar a captar el ojo humano está alrededor de los 200 FPS, aunque si bien es cierto que, tal y como se utilizaba en el cine, a partir de los 24 FPS el ojo humano ya no distingue el cambio de imágenes, sino que lo detecta como un movimiento continuado. Así pues el sistema de control deberá de ser capaz de generar y enviar señales a más de 24 Hz.

Por otra parte, el sistema de comunicación también tendrá dentro de sus requisitos la frecuencia, ya que deberá de ser capaz de recibir y enviar dichas señales a la misma velocidad mencionada anteriormente, de modo que no se pierda ninguna. Pero además también deberá de ser capaz de mantener la comunicación a una distancia nunca inferior a los 3 metros, para facilitar el uso del mando de control remoto, ya que de otro modo perdería prácticamente toda su comodidad.

Para finalizar, el sistema de recepción de datos deberá de ser capaz de recibir las señales a una frecuencia también superior a los 24 Hz y además deberá de ser capaz de reproducir en la pantalla del dispositivo Android imágenes a una velocidad igual o superior a los 24 FPS, de modo que el ojo humano no sea capaz de percibir en la pantalla una sucesión de imágenes, sino un movimiento continuado.

Una vez analizados todos los requisitos, para el proyecto se propusieron los siguientes componentes: Para el sistema de control se propuso un Joystick, formado por dos potenciómetros, capaz de generar las señales a más de 24Hz, el cual estaría conectado al microprocesador Arduino Uno, que también es capaz de tratar la señal y enviarla a más de 24 Hz. Precisamente para enviar la señal de utiliza el módulo de comunicación HC-05, módulo que cumple con creces los requisitos que se pedían al sistema de comunicación, ya que es capaz de recibir y enviar los datos a más de 24 Hz y además tiene un alcance de unos 5-10 metros de distancia. Finalmente, para el sistema de recepción de datos se ha diseñado una aplicación para Android mediante el programa MIT App Inventor 2, el cual permite introducir un timer de la frecuencia deseada, cumpliendo así con el último requisito indispensable del proyecto y dando el visto bueno a la implementación de la solución propuesta.

Figura 10. Conjunto de sistemas del proyecto

3.2 Análisis de alternativas para los diversos sistemas del proyecto.

3.2.1 Opciones para el sistema de control.

El sistema de control del proyecto se encuentra claramente diferenciado en dos partes distintas, por un lado, se tiene la generación de señal, que será la interfaz con la que interactuará la persona para reproducir el movimiento que quiera ver en la pantalla del smartphone, y por otro lado se tiene el tratamiento de dicha señal, el cual se realizará mediante un microprocesador. Se va a ver más a fondo cada una de esas partes a continuación.

3.2.1.1 Sistema de control: Generación de señal.

Dentro del sistema de generación de señal hay que tener en cuenta varios requisitos, entre los cuales destaca la comodidad para el usuario, ya que si el sistema de generación de señal es incómodo o dificulta su propio uso será difícil que el proyecto final destaque en ningún mercado.

Otro requisito indispensable en el diseño del sistema de generación de señal es que ha de ser capaz de generar señales para generar movimiento en 2D, es decir, ser capaz de generar dos

señales simultáneamente, una para el eje de ordenadas, y otro para el eje de abscisas, y ser capaz de alterar dichas señales tanto de manera creciente como de manera decreciente, de modo que pueda alterar la posición de un determinado objeto y colocarlo en cualquier posición dentro del plano 2D que representa una pantalla de smartphone.

Por último, pero no menos importante, otro de los requisitos sería que la frecuencia a la que se generan las señales sea suficientemente buena como para no notar un retardo en la señal, porque tal y como vimos anteriormente, existe una frecuencia a partir de la cual el ojo humano detecta la imagen como una imagen en movimiento y no como una secuencia de fotogramas, por lo tanto, para conseguir que el mando de una sensación de control sobre el movimiento de un objeto se necesita que el sistema generador de señales sea capaz de generar las señales a una frecuencia capaz de trabajar sin retardos, y permitir que en el Smartphone se reproduzca a una frecuencia nunca inferior a los 24 FPS.

Una vez analizados todos los requisitos necesarios para diseñar un generador de señales adecuado, se procederá a comparar varias opciones viables y se elegirá la mejor opción.

3.2.1.1.1 Análisis de opciones para el generador de señales: Botonera de 4 botones.

La primera opción que apareció encima de la mesa a la hora de escoger un generador de señal fue la de utilizar 4 botones, uno para incrementar la variable de la posición X, otro para decrementar dicha variable, un tercero para incrementar el valor de la posición en el eje Y, y finalmente el último para decrementar dicho valor.

Una vez contemplada esta solución se pasó a analizar el comportamiento de los botones respecto a los requisitos mencionados anteriormente, dónde se pudo observar que no habría problema con la frecuencia, ya que el pulsador trabaja a una frecuencia suficientemente alta como para no notar retardos entre el pulsado del botón y el movimiento del objeto en el Smartphone,

Para analizar la comodidad se realizó un estudio de los mandos de control remoto que existen en la actualidad, dónde obviamente destacan los mandos de control de videoconsolas, como la Play Station 4 o la X-Box 360 Pro.

Dentro de los mandos de control de videoconsolas del mercado, para poder encontrar uno que utilizase 4 botones para el control de la posición hay que remontarse a finales de los años 90.

En 1987 Nintendo lanzó al mercado el mando más conocido de todos los que contaban con 4 botones para controlar el movimiento, el mando de control de Nintendo ofrecía un pad direccional en forma de cruceta negra que se conectaba al sistema por medio de un puerto de 7 pines, y, acorde a los juegos de la época, más complejos y con más opciones, presentaba cuatro botones: dos rectangulares finos, Select y Start, y dos redondos y rojos, A y B.

Figura 11. Mando con control PAD de 4 botones

Este tipo de control de dirección predominó en el mercado durante varios años y muchas compañías fueron las que desarrollaron sus mandos de control siguiendo estas directrices, incluso en 1994 Play Station lanzaba su primer mando siguiendo este mismo tipo de control de movimiento. El PS Control Pad, como fue bautizado el mando, introdujo los famosos botones cuadrado, triángulo, círculo y X, tan representativos de PlayStation aún más de 20 años después.

Figura 12. Mando PlayStationcon PAD 4 botones

Tan solo 3 años después, en 1997, Sony actualizó sus mandos añadiendo dos joysticks que podían ser manejados cómodamente con los pulgares. Además, alteró mínimamente la carcasa del mando de control, alargando las agarraderas. Los joysticks tenían un diseño ergonómico que contenía unas hendiduras en la parte superior del propio joystick en las que el pulgar se podía apoyar cómodamente.

Figura 13. Mando PlayStation con Joystick

Así fue como el control de movimiento por Joystick fue sustituyendo a la botonera progresivamente hasta llegar a hoy en día donde prácticamente es imposible encontrar un mando en el mercado competitivo de los controles de consola que todavía trabaje con un Pad de 4 botones, utilizando todos el sistema de control de posición por joystick, utilizando habitualmente dos sticks, uno para el control de movimiento y otro para el control de la cámara.

3.2.1.1.2 Análisis de opciones para el generador de señales: Joystick.

Así pues, llegamos a la opción elegida para el control de posición del proyecto, el joystick.

Como ya se vio en el punto anterior, el diseño ergonómico del joystick está hecho para dar una sensación de comodidad al usuario, y para finalizar con el análisis de cumplimiento de los requisitos, se ha comprobado que la velocidad de generación de señal del joystick es lo suficientemente buena como para no notar ningún tipo de retardo entre el movimiento del joystick y el movimiento del objeto representativo en el Smartphone, así que, analizando todas las alternativas posibles y buscando la mejor opción respetando siempre el cumplimiento de los requisitos, se ha llegado a la conclusión que la mejor opción para el generador de señales es el Joystick.

Figura 14Mando X-Box de última generación

Figura15. Mando PlayStation última generación

3.2.1.2 Sistema de control: Tratamiento de la señal.

Para la elección del sistema de tratamiento de señal también se tuvieron en cuenta una serie de requisitos, entre los cuáles destacarían, por un lado, la velocidad de trabajo de los microcontroladores, aunque con los microprocesadores de hoy en día, en ningún caso la velocidad de procesamiento resultó ser un problema, por otro lado, el coste que supondría la compra de los microcontroladores y de su software correspondiente, y por último, la facilidad para programar y compilar código, unido a la existencia de una comunidad de programadores capaz de prestar ayuda y aportar ideas

Teniendo presente todos los requisitos mencionados anteriormente se procedió a comparar y escoger la opción más adecuada para el sistema de tratamiento de señal.

3.2.1.2.1 Análisis de opciones para el tratamiento de la señal: STM32f4

La primera opción que se estudió para el sistema de tratamiento de la señal fue el microcontrolador STM32F4 Discovery, porque ya se conocía el funcionamiento de dicha placa. En un principio la STM32F4 destacó sobre las demás opciones porque era mucho más potente que sus competidores y su mundo de posibilidades era mucho más amplio.

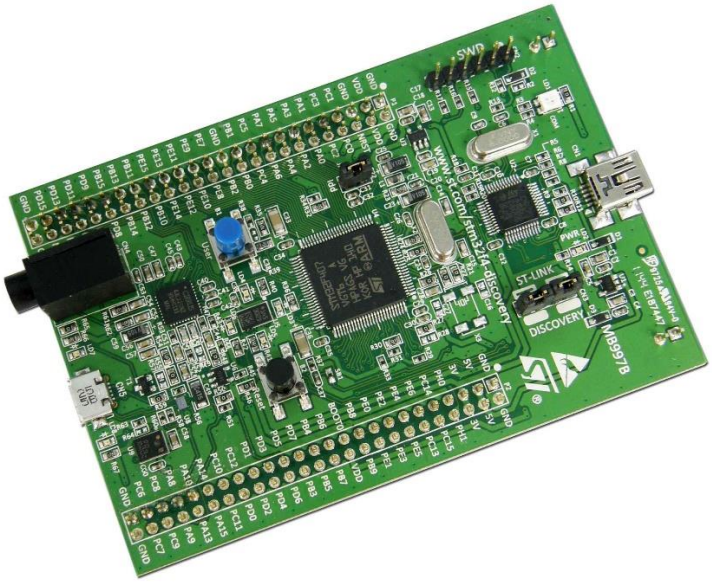


Figura 16. STM32F4 DISCOVERY

Aun contando con esta gran ventaja, la opción de usar una STM32F4 rápidamente se diluyó, al resultar ésta más difícil de encontrar en el mercado, y por lo tanto más cara, además de que el software que se necesitaba utilizar para programar este microcontrolador no era de acceso gratuito mientras que sus competidores si contaban con un software a coste cero.

Figura 17. Software de STM32F4 DISCOVERY

Figura 18. Logotipo de Keil uVision 5

También se tuvo en cuenta que no contaba con ninguna comunidad de programadores en la red capaces de prestar ayuda en momentos de necesidad.

3.2.1.2.1 Análisis de opciones para el tratamiento de la señal: Arduino UNO

Otra de las opciones estudiadas para el sistema de tratamiento de señal fue el microcontrolador Arduino UNO, debido a su gran fama. Rápidamente en comparación con el microcontrolador STM32F4 se pudo observar que tenía un potencial bastante menor, pero más que suficiente para el proyecto que se pretendía llevar a cabo, así que se hizo un estudio más exhaustivo de sus características, para analizar así su viabilidad.

Figura 19.Microccontrolador Arduino UNO

La opción de utilizar este microcontrolador cobró fuerza gracias a su fama, ya que resultó mucho más sencillo encontrarla en el mercado que a sus competidores, y por lo tanto, tenía un precio más asequible. Para terminar de decantar la balanza, el software necesario para programar Arduino UNO resultó ser de software libre, y al contrario que las demás opciones estudiadas, no implicaba un coste adicional.

Figura 20. Software de Arduino UNO

Figura 21. Icono de Software

Además de las ventajas vistas anteriormente, otra gran razón para decantarse por Arduino Uno es que cuenta con una gran comunidad online de programadores que respaldan a la plataforma y tienen la capacidad de dar consejo y/o prestar ayuda en momentos clave.

3.2.2 Opciones para el sistema de comunicación.

El sistema de comunicación del proyecto consta de unos requisitos básicos indispensables sin los cuales el correcto funcionamiento del proyecto sería inviable. Dichos requisitos serían, por un lado, que el gasto de energía que supondría para el sistema sea el mínimo posible, de manera que el proyecto final obtenga la máxima autonomía posible, y por otro lado, que la distancia máxima de comunicación se encuentre entre unos 3-5 metros, para poder proporcionar una mayor comodidad de uso. En cuanto al requisito de la velocidad de transmisión de datos ninguna de las opciones estudiadas ha dado ningún tipo de problema. Se van a analizar las dos opciones estudiadas a continuación.

3.2.2.1 Análisis de opciones para el sistema de comunicación: WiFi

La primera opción que se estudió para el sistema de comunicación fue la popular tecnología de comunicación inalámbrica WiFi. Cuando un dispositivo se encuentra habilitado con WiFi puede conectarse a otros dispositivos o a internet mediante un punto de acceso de red de comunicación inalámbrica (También conocido como WAP, por sus siglas en inglés, Wireless Access Point).

Figura 22.Logotipo de WIFI

Esta tecnología de comunicación inalámbrica destaca por su gran capacidad de ancho de banda, llegando a ser de hasta 1 Gbps, por su seguridad, y en último lugar por su rango, el cual alcanza hasta los 300 metros, pero esta última ventaja se ve afectada por uno de los grandes problemas de la tecnología wifi, que es la progresiva saturación del espectro radioeléctrico, motivado por la masificación de usuarios; que afecta a las conexiones superiores a los 100 metros de distancia y haciéndolas falibles. Por eso sólo sería aconsejable utilizarlo para usos domésticos, siempre inferiores a los 100 metros de distancia, evitando así estar expuesto al excesivo riesgo de interferencias.

En cuanto a seguridad, las redes WiFi presentan una seguridad que puede ser vulnerada, y, a pesar de que con una correcta configuración se evitarían la mayoría de los conflictos en cuanto a seguridad se refieren, lo cierto es que una gran cantidad de redes WiFi no son configuradas adecuadamente y terminan siendo vulnerables a ataques de terceros. De todos modos, en el proyecto no será de vital importancia la seguridad del sistema de comunicación y por lo tanto no se prestará más atención a ello.

Por último, una de las cualidades que se han estudiado a la hora de elegir el sistema de comunicación ha sido el gasto que supondría implantar dicho tipo de sistema de comunicación. En el mercado se ha encontrado un módulo de comunicación WiFi, que sería el Módulo ESP8266, por un coste final inferior a los tres euros (€).

Figura 23. Módulo ESP8266

3.2.2.2 Análisis de opciones para el sistema de comunicación: Bluetooth

La gran alternativa a la comunicación inalámbrica WiFi fue la tecnología de comunicación Bluetooth. Bluetooth presentó una capacidad de ancho de banda muy inferior a la tecnología WiFi, siendo solamente de 24 Mbps. También presento una seguridad más baja, y un rango de comunicación muy inferior al de la tecnología WiFi, alcanzando solamente una distancia de 30 metros, como máximo. Además, mientras en la tecnología WiFi se permite trabajar a distintas frecuencias (2.4 GHz, 3.6 GHz, ó 5 GHz), en la tecnología Bluetooth sólo se puede trabajar con la frecuencia de 2.4 GHz.

Por último, pero no menos importante, con la tecnología WiFi se tiene la posibilidad de establecer una red de dispositivos, es decir, tener un conjunto de equipos conectados de forma simultánea, mientras que con la tecnología Bluetooth sólo se puede emparejar los dispositivos por pares.

A pesar de todos estos inconvenientes, ninguno de ellos supone un problema serio para el proyecto, ya que por un lado, no se necesitan enviar paquetes de datos demasiado grandes, así que el ancho de banda de Bluetooth de 24 Mbps es más que suficiente. Por otro lado, la seguridad no es un problema en nuestro proyecto, ya que no el prototipo del mando inalámbrico de control remoto llevará una contraseña de conexionado Bluetooth predefinida evitando posibles robos de información, además de que para interactuar con el sistema deberá estar a unos 5-10 metros de distancia del dispositivo.

También se ha tenido en cuenta que el rango de 30 metros cumple de sobra con las necesidades expuestas anteriormente, y trabajar a 2.4 GHz no supone ningún problema ni

complicación en la realización del proyecto.

Finalmente, el emparejamiento de dispositivos por pares se adapta perfectamente a las necesidades del proyecto y no causaría ningún problema.

Una vez se comprobó que ninguna de las desventajas que presentaba la tecnología Bluetooth frente a la tecnología WiFi causaría ningún problema para el proyecto, se analizaron sus ventajas, siendo principalmente una, el gasto energético.

La tecnología inalámbrica WiFi tiene un consumo de energía mucho más elevado que en el caso del Bluetooth, y más en la actualidad, cuando se lanzó un nuevo modelo de Bluetooth 4.0 L.E. (Proviene de las siglas en inglés, Low Energy) el cuál alcanza una eficiencia energética muy alta en comparación con la tecnología WiFi.

Figura 24. Bluetooth 4.0 L.E.

Así pues, una vez se decidió que la tecnología Bluetooth era la que mejor se adaptaba al proyecto, se hizo un estudio de mercado en busca de los módulos de comunicación Bluetooth que supusieran un gasto conforme al resto del proyecto y fueran perfectamente válidos, encontrando en dicho estudio el módulo de bluetooth integrado HC-05 por un precio inferior a los cuatro euros (€), un coste acorde al proyecto.

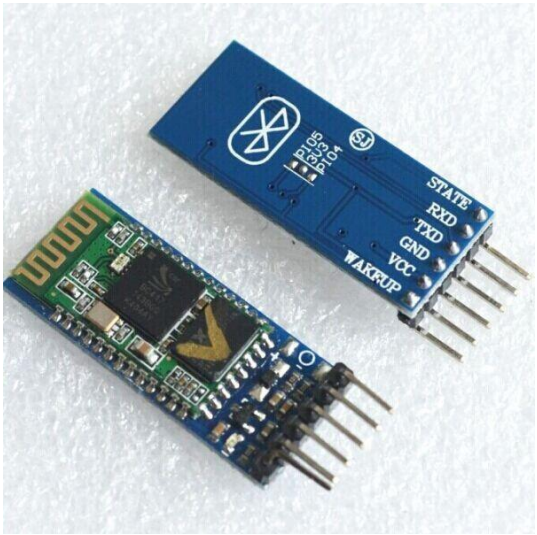


Figura 25. Modulo Bluethooth HC-05

WiFi y Bluetooth: Diferencias principales

	Bluetooth	WiFi
Frecuencia	2.4 GHz	2.4 / 3.6 / 5 GHz
Ancho de banda	24 Mbps	1 Gbps
Seguridad	Baja	Moderada
Rango	hasta 30 metros	hasta 300 metros
Consumo	Reducido	Elevado

Tabla 3. Características de Bluetooth VS WiFi

3.2.3 Opciones para el sistema de alimentación.

El sistema de alimentación del proyecto ha de ser adecuado para dar al sistema una cierta autonomía y dotarlo de comodidad, por ello no se utilizará ningún sistema de conexión por cable. Para el sistema de alimentación tendremos dos opciones principalmente, por un lado, podríamos emplear un regulador de voltaje integrado en la placa, y por otro podríamos aplicar directamente el voltaje regulado a la tensión nominal de la placa.

La primera de las opciones es viable gracias a que el microcontrolador Arduino UNO, al igual que todos los microcontroladores de Arduino, dispone de un regulador de voltaje integrado. Dicho regulador conlleva una pequeña caída de voltaje, por lo que habría que aplicar al menos una tensión de 6 Voltios, para evitar que baje de los 5 Voltios y se apague. En el otro extremo se tendría el exceso de voltaje, el cual nunca debería superar los 12 Voltios, ya que por encima de ellos el disipador tendrá que disipar mucho más calor y conllevaría un esfuerzo innecesario.

En la segunda opción se pueden aplicar directamente una tensión nominal de 5 Voltios, que es lo mismo que se hace cuando se conecta Arduino mediante el puerto USB, pero sería prioritario tener una fuente de alimentación regulada, ya que cualquier pico de tensión podría dañar la placa de manera irreparable.

3.2.3.1 Análisis de opciones para el sistema de alimentación: Pilas eléctricas.

La primera opción que se ha estudiado a la hora de elegir un sistema de alimentación ha sido la posibilidad de utilizar las clásicas pilas eléctricas. Existen pilas de distintos tipos según su uso, las más utilizadas son las de 1,5 Voltios, que son las más comunes para uso doméstico, las de 6 Voltios, que se utilizan rara vez en mandos de control remoto, y las de 12 Voltios, que se suelen utilizar en alarmas de viviendas.

Figura 26.Pila eléctrica

En un principio las pilas alcalinas no suelen ser demasiado caras, y su precio oscila alrededor de unos 3-5 euros (€), lo cual entraría dentro del presupuesto global del proyecto, pero, al no ser recargables, a largo plazo no resultan económicas, y teniendo en cuenta la comodidad del cliente, no son la opción preferente, debido a que cada poco tiempo el usuario que haya adquirido el mando tendrá que preocuparse por reemplazar las pilas ya agotadas. Por estos motivos se ha optado por buscar otra opción más viable a la hora de elegir el sistema de alimentación del proyecto, pero si se quisiera elegir éste, se debería de añadir un porta pilas de cuatro pilas, colocadas en serie, de voltaje 1,5 Voltios cada pila. Las cuatro pilas proporcionarían unos 800-1500 mAh en caso de ser pilas AA convencionales, y si fueran pilas AA alcalinas llegarían hasta los 1700-2800 mAh. El coste del porta pilas no sobrepasaría el euro (€).

Figura 27. Ejemplo de porta pilas

3.2.3.2 Análisis de opciones para el sistema de alimentación: Banco de Batería USB.

En la actualidad el uso de los bancos de batería USB (Más conocidos como Power Bank, de su nombre en inglés) está ampliamente extendido principalmente para alargar la duración de la batería de los smartphones, pero resultarían perfectamente viables para alimentar el microcontrolador del proyecto.

Los bancos de batería USB proporcionan un voltaje de 5 Voltios perfectamente regulado, por lo que no habría ningún tipo de problemas a la hora de alimentar el Arduino UNO mediante su puerto USB, sin el requisito de regular la tensión.

Dicho voltaje es idóneo para alimentar la mayoría de los componentes utilizados en el proyecto, ya que tanto el módulo Bluetooth como el Joystick operan bajo esa tensión.

Finalmente se ha analizado el coste que supondría al cliente la compra de una Power Bank, y a pesar de que son bastante más caras que las pilas (Se pueden encontrar desde unos 10 euros hasta unos 50 euros si se buscan las que pueden almacenar más carga), el hecho de que sean recargables hace de ellas una inversión más rentable a largo plazo, sin olvidar que pueden ser utilizadas en distintos dispositivos, y el usuario las podrá emplear situacionalmente en su teléfono móvil.

Figura 28. Batería externa

3.2.4 Opciones para el sistema de recepción.

En cuanto a la elección del sistema de recepción se barajaron varias opciones totalmente distintas. Al principio se trató de contactar con las compañías poseedoras de los derechos de los videojuegos de telefonía móvil más exitosos del momento, en concreto con Niantic Labs y con Supercell, dueñas de los ya famosos videojuegos Pokémon Go y Clash Royale, respectivamente. La idea que se tenía sobre la mesa era la de utilizar el proyecto en relación con sus videojuegos durante la presentación del proyecto, pero ninguna de ambas se prestó.

Figura 29. Logotipo Niantic

Figura 30. Logotipo de Supercell

Niantic argumentó que la política de control de posición de personaje mediante joystick incumplía los términos de servicio (ToS, de sus siglas en inglés Terms of Service) y que incumplir dichas políticas para un proyecto universitario conllevaría a sembrar un precedente entre sus seguidores que podría acarrearle problemas a largo plazo.

Por su parte Supercell respondió que la jugabilidad de su producto se vería enormemente alterada, ya que en un principio el videojuego fue diseñado para ser utilizado mediante un panel táctil, y esto conllevaría a una opinión desfavorable del usuario. A pesar de los numerosos esfuerzos y la gran cantidad de argumentos que se le presentaron se negó rotundamente a ceder los derechos para el acto de presentación del proyecto.

Así pues, sin el respaldo de ninguna de las dos compañías punteras, se tuvo que optar por una idea mucho menos vistosa, pero que conllevaba un gran esfuerzo igualmente, que era desarrollar una aplicación que sirviera a su vez tanto para recibir los datos como para hacer una demostración interactiva del proyecto, es decir, se diseñó una aplicación que serviría de receptora de datos y al mismo tiempo sería un videojuego.

Con la finalidad de encontrar un entorno de trabajo adecuado para dicho fin, se hizo un estudio de mercado de los mejores lenguajes de programación, principalmente orientados a objetos, de manera que resultase viable realizar una aplicación de Android que cumpliera los requisitos

establecidos anteriormente. En el estudio realizado destacaron dos opciones perfectamente viables, la primera, MIT App Inventor 2, se trataba de un entorno de desarrrollo de software diseñado especialmente para facilitar la creación de aplicaciones Android. La plataforma trabaja con Java, pero el usuario simplemente se servirá de un conjunto de herramientas básicas proporcionadas por App Inventor para enlazar bloques creando así la aplicación. Si bien este entorno era más que suficiente para desarrollar la aplicación buscada, en un principio se optó por buscar otra opción más potente. En este marco apareció la segunda opción, APDE Processing (De sus siglas en inglés Android Processing Development Environment), una aplicación basada en Processing, un lenguaje de programación y entorno de trabajo basado en Java, sencillo, pero a su vez más potente de App Inventor. Cabe destacar que ambas opciones eran de código abierto, lo cual permitía la existencia de una comunidad activa que pudiera respaldar el proyecto.

Figura 31. APDE

3.2.4.1 Análisis de opciones para el sistema de recepción: APDE Processing

Processing fue desarrollado en 2001 en el MIT, obra de Casey Reas y Ben Fry, basándose en el DBN(De sus siglas en inglés, Design By Numbers), obra de John Maeda. Una ventaja de Processing era que permitía trabajar tanto en una terminal Android simulada en un computador, como en un teléfono móvil directamente, gracias a su aplicación en Android APDE, mientras que en App Inventor era indispensable el uso de un ordenador.

Processing parte de dos funciones básicas, la setup, que se ejecuta al iniciar el programa, y es dónde vienen recogidas las instrucciones que se desean ejecutar una sola vez, y la función draw, que se ejecuta continuamente, y es donde vienen recogidas las instrucciones que se pretenden repetir una y otra vez. En la cabecera del principio se pueden importar las librerías bajo la llamada import, y las funciones se definen justo debajo. Se verá más claramente con una imagen.

sketch

```
import controlP5.*;

int uX, uY;

void setup()
{
  size (1050,1900);
  uX=width/18; //Valores para que se
       //muestre correctamente en la
       //pantalla del smartphone.
  uY=height/32;
}

void draw()
{

}
```

Una vez ya más adentrados en lo que sería la programación el Processing, se empezaron a utilizar las clases, y se hizo uso de algunas funciones para diseñar algunas aplicaciones de prueba. Se va a ver ahora un código de una aplicación que hace que la pantalla del smartphone cambie de color al deslizar el dedo por un control deslizante (Slider a partir de ahora) para poder ver como se usan las clases y las funciones mencionadas anteriormente.

Slider

```
import controlP5.*;

int uX, uY;
ControlP5 Cp5;

void setup()
{
  size (1050,1900);
  uX=width/18;
  uY=height/32;
  Cp5= new ControlP5(this);
  Cp5.addSlider("Tonalidad").setPosition(4*uX,3*uY)
                            .setSize(10*uX,26*uY)
                            .setRange(0,255)
                            .setValue(0)
                            .setColorValueLabel(color(0))
                            .setColorCaptionLabel(color(255));
  Cp5.getController("Tonalidad").getCaptionLabel().align(ControlP5.CENTER,ControlP5.BOTTOM_OUTSIDE).setPaddingX(0);
  Cp5.getController("Tonalidad").getValueLabel()
                                .setFont(createFont("",50));
  Cp5.getController("Tonalidad").getCaptionLabel()
                                .setFont(createFont("",50));


}

void draw()
{

}

void Tonalidad (float ValorRango)
{
  background(255-(ValorRango/2), ValorRango,255-ValorRango);

}
```

Finalmente se llegó a un punto en el cual se tenía un dominio suficiente de Processing como para desarrollar una aplicación que realizara todos los requisitos que se vieron al principio, pero desgraciadamente la aplicación tenía una librería obsoleta para la conexión del bluetooth en la gama de modelos de telefonía móvil en el cual se iba a probar el proyecto, de modo que el programa quedó en algo meramente teórico, a pesar de funcionar perfectamente en la simulación.

3.2.4.2 Análisis de opciones para el sistema de recepción: MIT App Inventor 2.

Finalmente se decidió realizar la aplicación encargada de recibir los datos y interactuar con el usuario utilizando en entorno de desarrollo MIT App Inventor 2.

MIT App Inventor tiene un entorno de trabajo más sencillo que APDE Processing, y mucho más intuitivo. Su uso se encuentra dividido en dos partes, la primera se trata de un área de trabajo para el diseño de la interfaz de usuario, en la cual se pueden añadir prácticamente todos los elementos que sean necesarios para el proyecto.

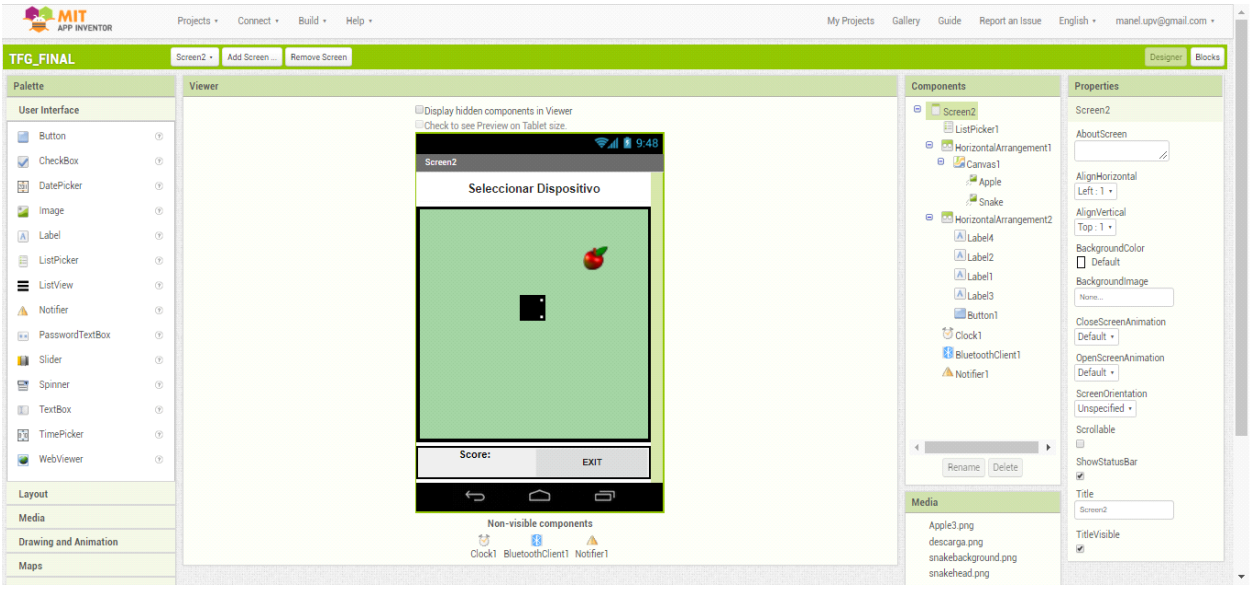


Figura 32. Interfaz gráfica de MIT App Inventor 2.

La segunda parte, dónde se programa todo el código de la aplicación, está diseñada de modo que las instrucciones vengan en forma de bloques, los cuales se unen como si fueran puzles, haciendo la programación muy sencilla. Los bloques están separados por colores según su tipo, los hay desde funciones numéricas (+, -, *, =, etc.) hasta instrucciones de control en forma de bucle (for, if, while, etc.).

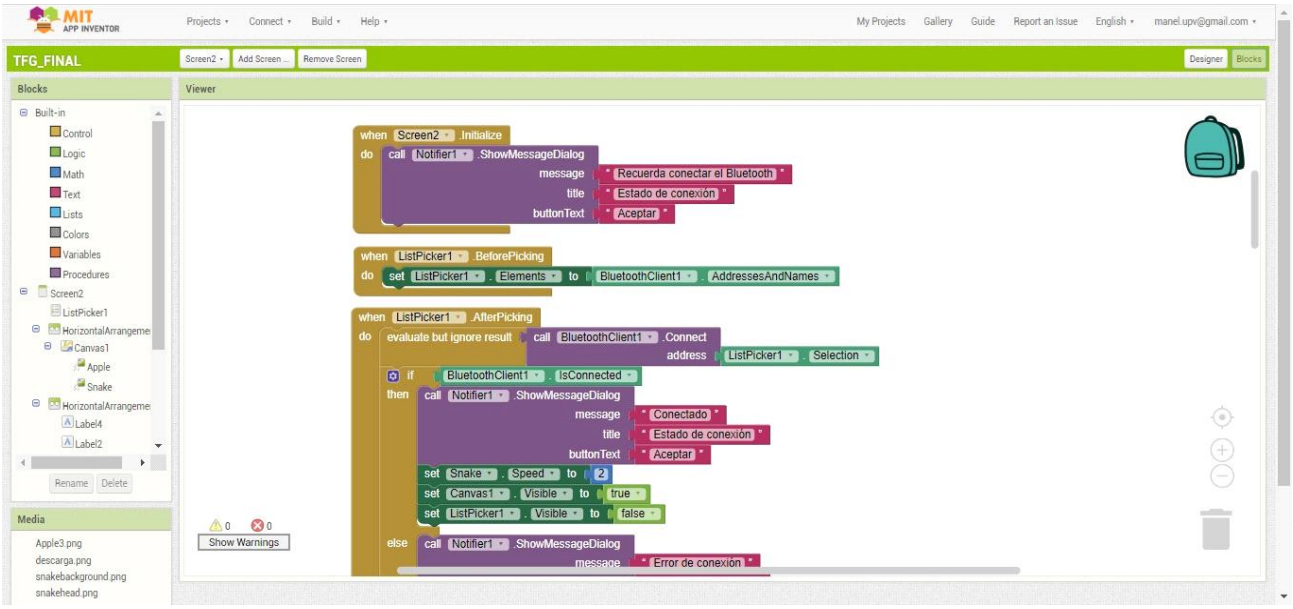


Figura 33 Interfaz de desarrollo de código de MIT App Inventor 2.

3.3 Desarrollo de las soluciones aplicadas al proyecto.

3.3.1 Desarrollo de la solución para el sistema de generación de señal.

En primer lugar, para generar la señal se ha escogido el joystick. Para comprender mejor el funcionamiento del joystick, se va a explicar brevemente su composición. El joystick está formado por dos potenciómetros, dónde cada uno de los cuáles se encargan de controlar una variable que representa la variable del objeto que se desea mover en uno de los ejes, así pues, se puede incrementar y decrementar el valor de posición del objeto respecto a un eje con cada potenciómetro, pudiendo así situar el objeto en cualquier punto del plano 2D que representa la pantalla del Smartphone con un solo Joystick.

La conexión del joystick sería la siguiente.

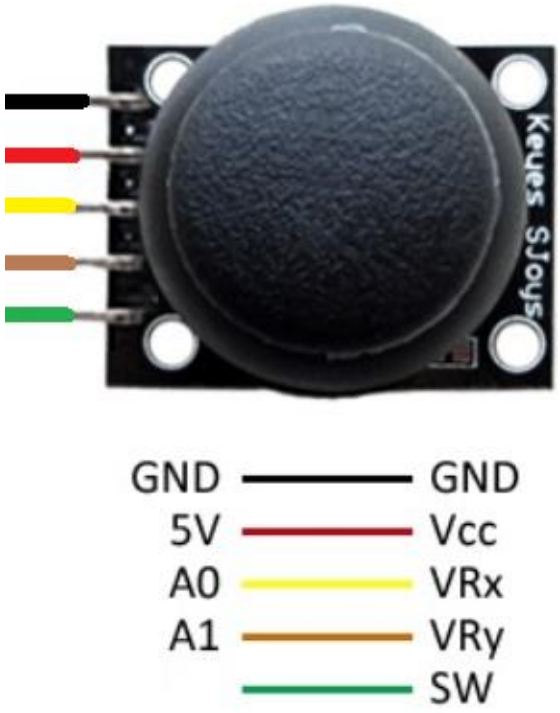


Figura 34. Conexionado del Joystick.

Como se puede observar el joystick tiene además la posibilidad de utilizar un pulsador (Switch) en la patilla de abajo, pero se ha decidido dejar sin conectar (al aire), ya que en un principio no se le va a dar uso al botón.

3.3.2 Desarrollo de la solución para el sistema de tratamiento de señal.

A pesar de que el microcontrolador STM32f4 no fue seleccionado para el proyecto, sí se logró realizar un código que cumpliera con la función deseada, y aunque no se llegue a utilizar a lo largo del proyecto sí se considera interesante dejar constancia sobre él, por lo tanto, se mostrará a continuación.

Para empezar, se incluyen las librerías en el proyecto.

```
#include <stdio.h>
#include "stm32f4xx_rcc.h"
#include "stm32f4xx_gpio.h"
#include "stm32f4_discovery.h"
#include "stm32f4xx_adc.h"
#define VOLT_REF     2950       /* Referencia de Voltaje en mV */
```

A continuación, se configuran los puertos de entrada, hay que comprobar con anterioridad que no haya nada conectado internamente a los pines que vayan a ser utilizados para configurar el ADC, en este caso se ha comprobado que el ADC puede ser conectado a los pines A0 y A1, estando configurado el ADC1 en el pin A1.

```
void config_entradas(void){
        GPIO_InitTypeDef GPIO_InitStructure;
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1 | GPIO_Pin_0;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN;
//GPIO_InitStructure.GPIO_OType = GPIO_OType_PP; //No es necesario
        GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
//GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz; //No es necesario
        GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

Siguiendo con un programa bien estructurando lo siguiente que se hace es configurar el ADC.

```
void config_adc(void){
ADC_InitTypeDef ADC_InitStructure;
ADC_CommonInitTypeDef ADC_CommonInitStructure;
//RCC_RTCCLKConfig(RCC_HCLK_Div4); //No es necesario
RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);

ADC_CommonStructInit(&ADC_CommonInitStructure);
ADC_CommonInitStructure.ADC_Mode = ADC_Mode_Independent;
ADC_CommonInitStructure.ADC_Prescaler = ADC_Prescaler_Div4; // max 30 MHz
ADC_CommonInitStructure.ADC_DMAAccessMode = ADC_DMAAccessMode_Disabled;
ADC_CommonInitStructure.ADC_TwoSamplingDelay = ADC_TwoSamplingDelay_5Cycles;
ADC_CommonInit(&ADC_CommonInitStructure);

ADC_StructInit(&ADC_InitStructure);
ADC_InitStructure.ADC_Resolution = ADC_Resolution_12b; /*La resolución puede
ser 12, 10, 8 y 6 bits*/
ADC_InitStructure.ADC_ScanConvMode = DISABLE; /*Conversión de dos canales */
ADC_InitStructure.ADC_ContinuousConvMode = DISABLE; /*Tipo de conversión en m
odo continuo*/
ADC_InitStructure.ADC_ExternalTrigConvEdge = ADC_ExternalTrigConvEdge_None; /
*No utilización de disparo externo para realizar la conversión*/
ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
        ADC_InitStructure.ADC_NbrOfConversion = 1;
        ADC_Init(ADC1,&ADC_InitStructure);
         ADC_Cmd(ADC1, ENABLE);
}
```

Siguiendo con el código, lo siguiente que se realiza es una función que realice la conversión del valor analógico que proporciona el joystick y se pasa a un valor digital, que será el que devuelve la función.

```
int posicion_x(void)
{
uint32_t v;
uint32_t value;
ADC_InjectedSequencerLengthConfig(ADC1, 1);
ADC_SetInjectedOffset(ADC1, ADC_InjectedChannel_1, 0);
/* ADCx injected channel Configuration */
ADC_InjectedChannelConfig(ADC1, ADC_Channel_1, 1, ADC_SampleTime_15Cycles);
ADC_ClearFlag(ADC1,ADC_FLAG_JEOC);
ADC_SoftwareStartInjectedConv(ADC1);

while (ADC_GetFlagStatus(ADC1,ADC_FLAG_JEOC) == RESET);
         v = ADC_GetInjectedConversionValue(ADC1, ADC_InjectedChannel_1);
         value = (v * VOLT_REF)/0xFFF; // 12 bits ADC
         return v;
}
```

Para terminar el desarrollo del programa se va a mostrar ahora el cuerpo principal del programa.

```
int main(void)
{
int posX;
int posY;
int i;
config_entradas();
config_adc();
while(1){
         posX = posicion_x();
         posY = posicion_y();
         printf("posicion eje X - %dn", (int) posX);
         printf("posicion eje Y - %dn", (int) posY);

         for (i=0;i<10000000;i++); //Es una espera para que de tiempo a ver el valor.
         }
         return(0);
}
```

Ahora se va a ver y comentar el código de Arduino Uno, que fue utilizado para llevar a cabo el proyecto cumpliendo con todas las funciones deseadas.

Para comenzar se incluyen las librerías y se definen los parámetros básicos y las variables a utilizar. Con los parámetros básicos se asigna tanto el Receptor (Rx) como el Transmisor (Tx) a los pines 10 y 11 de Arduino Uno respectivamente, para después conectar el módulo de comunicación que se explicará más adelante.

```
#include <SoftwareSerial.h>   // Se incluye la librería  SoftwareSerial

#define RxArduino 10
#define TxArduino 11
#define PinX A0
#define PinY A1

int X,Y;
char buffer[10]; // Se define un vector de tamaño 10
int ValorX;
int ValorY;
int boton=3;
SoftwareSerial BT(RxArduino,TxArduino); // Se define los pines RX y TX conectados al Bluetooth
```

Siguiendo la estructura básica de un programa de Arduino, aparece el setup del código, qué es el lugar donde se programan las funciones que se desean se ejecuten primero, puesto que el setup es la primera función en ejecutarse dentro del programa. Dentro del setup se establecen los criterios que solo requieren de una ejecución única, como asignar un determinado pin a salida o entrada de voltaje, utilizando el comando pinMode OUTPUT o INPUT respectivamente, indicándole así a Arduino cómo funcionará dicho pin.

```
void setup()
{
  BT.begin(9600);       // Se inicializa el puerto serie BT (Para Modo AT 2)
  Serial.begin(9600);   // Se inicializa  el puerto serie
  pinMode(PinX,INPUT);
  pinMode(PinY,INPUT);
  pinMode(boton,INPUT);
}
```

Para finalizar, aparece el loop del programa, qué es el lugar del código dónde se programan las instrucciones que se quieren repetir infinitas veces. Es decir, cuando un código de Arduino empieza, lleva a cabo las asignaciones, después ejecuta una sola vez el setup, y después entra en un bucle infinito dónde realiza una y otra vez las instrucciones que estén dentro del loop.

```
void loop()
{
  ValorX=analogRead(PinX);
  ValorY=analogRead(PinY);
  X=map(ValorX,0,1023,0,99);
  Y=map(ValorY,0,1023,0,99);
 if(digitalRead(boton)==HIGH){
  sprintf(buffer, "%d, %d, 1", X, Y);
  Serial.println(buffer);
  BT.println(buffer);
  delay(200);
  }
   if(digitalRead(boton)==LOW){
  sprintf(buffer, "%d, %d, 0", X, Y);
  Serial.println(buffer);
  BT.println(buffer);
  delay(200);
  }
}
```

Analizando el código se puede observar como al principio se asignan a las variables ValorX y ValorY los valores analógicos de las lecturas del joystick del eje X y del eje Y respectivamente. Después, se mapean dichos valores analógicos, consiguiendo de esta manera tener unos valores de posición de X e Y que vayan entre 0-99, en lugar de ir de 0-1023, lo cual resultaría un poco incómodo a la hora de trabajar. Una vez se tienen los valores transformados se llega a una disyuntiva en la que se encuentran las siguientes opciones, o bien el botón asignado con anterioridad al pin 3 está pulsado, en cuyo caso se generará un vector de 3 valores, que sería el siguiente (ValorX, ValorY, 1), o bien el botón no está pulsado, en cuyo caso el vector de 3 valores generado sería (ValorX, ValorY, 0). Como se puede suponer, el 1 representa el pulsado del botón, y el 0 representa el no pulsado. Una vez el vector está generado lo único que falta por hacer es enviar dicho vector vía bluetooth y realizar una pausa de unos 0,2 segundos, para lo cual se ha utilizado la instrucción delay(200), que trabaja en milisegundos.

El motivo para utilizar 0,2 segundos es porque se ha llegado a dicho valor óptimo después de realizar un estudio a base de pruebas, para evitar un envío masivo de datos por un lado, y por otro evitar demasiada tardanza entre el envío de un dato y el envío de otro.

3.3.3 Desarrollo de la solución para el sistema de comunicación.

El módulo Bluetooth cuenta con 6 patillas, la última de las cuales (STATE) quedará al aire, puesto que no va a ser utilizada.

Las otras cinco con las que cuenta son la alimentación (Vcc), que se conectará a 5 V, la tierra (GND), que se conectará a la masa del microcontrolador, el enable (ENABLE), que se conectará a 3,3 V para obtener un correcto funcionamiento del módulo Bluetooth, y por último Rx y Tx, que se encargan se la recepción y la transmisión de datos respectivamente. Para ver esto de una manera más visual se tiene la siguiente imagen.

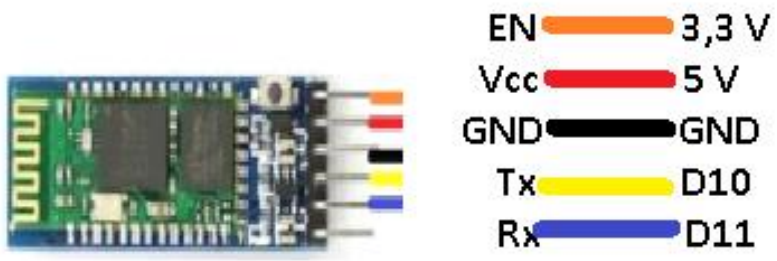


Figura 35. Conexionado del módulo Bluetooth.

3.3.4 Desarrollo de la solución para el sistema de alimentación.

Dentro del sistema de alimentación se ha escogido una power bank porque son sencillas de obtener y de utilizar, además de tener una vida útil muy superior a la de las pilas. El uso de la power bank no tiene mucho misterio, simplemente se conectará al microcontrolador mediante un tipo de cable denominado cable de conexión de tipo A a tipo B. Simplemente se comprobará que la power bank esté cargada antes de utilizarla y mediante el puerto de salida USB que tiene se conectará a Arduino utilizando el cable anteriormente mencionado.

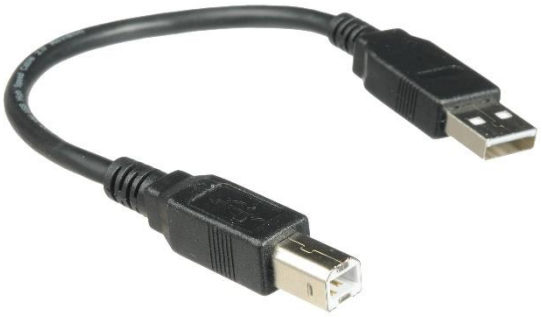


Figura 36. Cable de tipo A a tipo B.

3.3.5 Desarrollo del sistema de recepción.

Para evitar que todo el esfuerzo de desarrollar el código de la aplicación en processing fuese en vano, se va a mostrar el código completo de la aplicación en Processing que funcionaría si se dispusiera de otro modelo de smartphone, por si a alguien en un futuro lo intentase realizar.

Antes de mostrar el código se tratará de dar una breve explicación de lo que haría dicha aplicación. Por una parte, conectaría el Bluetooth de la terminal Android, para recibir así los datos que le llegarían procedentes del microcontrolador Arduino UNO, y por otra, dibujaría un punto rojo sobre un fondo negro que se movería con el movimiento del joystick, de manera que se pudiera controlar el punto rojo mediante el uso del mando de control remoto diseñado en el proyecto. Ahora vamos a ver el código separado por ventanas y se comentarán una a una.

El programa principal cuenta con 9 pestañas (Boton, Pressed, Dragged, Released, Bluetooth, Acciones, AreaDeDibujo, Joystick y Tabs2) que son en las pequeñas subpestañas en los que se divide el código principal.

Se van a exponer de manera ordenada para así facilitar su comprensión, ya que no se entrará en explicaciones demasiado detalladas al no tratarse del código final del proyecto.

Lo primero que se debe hacer al empezar a programar la aplicación es delimitar el área de dibujo sobre la que se va a trabajar, que se haría de la siguiente manera.

th Acciones AreaDeDibujo a17_Joystick

```
public class AreaDeDibujo
{
  float Xi,Xf,Yi,Yf;
  float Largo, Alto;
  color Area;

  AreaDeDibujo(float X, float Y, float L, float A, color Ca)
  {
    pushStyle();
    noStroke();
    Xi=X;
    Yi=Y;
    Largo=L;
    Alto=A;
    Xf=Xi+Largo;
    Yf=Yi+Alto;
    Area=Ca;
    fill(Area);
    rect(Xi,Yi,Largo,Alto);
    popStyle();
  }

  void Show()
  {
    pushStyle();
    noStroke();
    fill(Area);
    rect(Xi,Yi,Largo,Alto);
    popStyle();
  }


 void Hide()
 {
   pushStyle();
   noStroke();
   fill(g.backgroundColor);
   rect(Xi,Yi,Largo+1,Alto+1);
   popStyle();
 }
}
```

Una vez se tiene definida el área de dibujo lo siguiente que habría que hacer es definir las funciones de pulsado, arrastrado y soltado del botón. Dichas funciones servirán para detectar una interacción del ratón (En caso de usar un entorno de simulación de Android) o del dedo humano (En caso de trabajar directamente sobre el smartphone) con el botón de la interfaz gráfica. Para que el botón funcione correctamente son necesarias las funciones de pulsado, para detectar cuando se pulsa, de arrastrado, para detectar cuando se mantiene pulsado, y de soltado, para detectar cuando se termina de pulsar el botón. Dichas funciones se programarán de la siguiente manera respectivamente.

AreaDeDibujo Pressed

```
void mousePressed()
{
  Pestanas.Pressed();
  if(Boton1.Activo)
  Boton1.Pressed();
  if(Boton2.Activo)
  Boton2.Pressed();
}
```

AreaDeDibujo Dragged

```
void mouseDragged()
{
  if(Boton1.Activo)
  Boton1.Dragged();
  if(Boton2.Activo)
  Boton2.Dragged();
  }
}
```

AreaDeDibujo Released

```
void mouseReleased()
{
  Pestanas.Released();
  if(Boton1.Activo)
  {
    Boton1.Released();
    if(Boton1.vReleased)
    AccionesBoton1();
  }
  if(Boton2.Activo)
  {
    Boton2.Release();
    if(Boton2.vReleased)
    AccionesBoton2();
  }
}
```

Una vez se tienen definidas las funciones básicas del botón se puede programar directamente la creación de este, que sería de la siguiente manera.

Boton

```
  public class BotonRedondo
{
  int TamanoTextoBoton;
  int CentroX, CentroY, Diametro;
  color ColorTextoBoton;
  color ColorBoton, ColorBotonPressed;
  String TextoBoton;
  boolean vDragged, vReleased, Activo;

  BotonRedondo(int X, int Y, int D, color Cb, color Cp, String T, int Tt, color Ct)
  {
    CentroX= X;
    CentroY= Y;
    Diametro= D;
    ColorBoton= Cb;
    ColorBotonPressed= Cp;
    TextoBoton= T;
    TamanoTextoBoton=Tt;
    ColorTextoBoton=Ct;
    vDragged=false;
    vReleased=false;
    Activo=true;
    smooth();
    noStroke();
    fill(ColorBoton);
    ellipse(CentroX,CentroY,Diametro,Diametro);
    textSize(TamanoTextoBoton);
    fill(ColorTextoBoton);
    textAlign(CENTER, CENTER);
    text(TextoBoton,CentroX,CentroY);
    }

    void Hide()
    {
      Activo=false;
      fill(g.backgroundColor);
      stroke(g.backgroundColor);
      rect(CentroX-(Diametro/2)-1,CentroY-(Diametro/2)-1, Diametro, Diametro);
      noStroke();
    }

    void Show()
    {
      Activo=true;
      smooth();
      noStroke();
      fill(ColorBoton);
      ellipse(CentroX,CentroY,Diametro,Diametro);
      textSize(TamanoTextoBoton);
      fill(ColorTextoBoton);
      textAlign(CENTER, CENTER);
      text(TextoBoton,CentroX,CentroY);
    }
```

```
    void Pressed()
    {
      vReleased=false;
      if(dist(CentroX,CentroY,mouseX,mouseY)<Diametro/2)
      {
      smooth();
      noStroke();
      fill(ColorBotonPressed);
      ellipse(CentroX,CentroY,Diametro,Diametro);
      textSize(TamanoTextoBoton);
      fill(ColorTextoBoton);
      textAlign(CENTER, CENTER);
      text(TextoBoton,CentroX,CentroY);
      }
    }

    void Dragged()
    {
      if(dist(CentroX,CentroY,mouseX,mouseY)<Diametro/2)
      vDragged=true;
    }

    void Released()
    {
      if(dist(CentroX,CentroY,mouseX,mouseY)<Diametro/2)
      {
      smooth();
      noStroke();
      fill(ColorBoton);
      ellipse(CentroX,CentroY,Diametro,Diametro);
      textSize(TamanoTextoBoton);
      fill(ColorTextoBoton);
      textAlign(CENTER, CENTER);
      text(TextoBoton,CentroX,CentroY);
      vReleased = true;

    }

    if(vDragged)
    {
      vDragged=false;
      smooth();
      noStroke();
      fill(ColorBoton);
      ellipse(CentroX,CentroY,Diametro,Diametro);
      textSize(TamanoTextoBoton);
      fill(ColorTextoBoton);
      textAlign(CENTER, CENTER);
      text(TextoBoton,CentroX,CentroY);
      vReleased=true;
    }


}
}
```

Una vez se tiene tanto el botón como sus interacciones correctamente configuradas, lo siguiente que habría que hacer es asignar la funcionalidad de dicho botón, esto se realizaría mediante la pestaña Acciones.

Acciones

```
void AccionesBoton1()
{
  if(!Conectado)
  {
  AreaDeTexto.clear();
  BlueTooth.discoverDevices();
}
else
AlertaConectado();
}


void AccionesBoton2()
{
  if(!Conectado)
  {
  if (BlueTooth.getDiscoveredDeviceNames().size()>0)
  ListaKetai=new KetaiList(this, BlueTooth.getDiscoveredDeviceNames());
  else
  if(BlueTooth.getPairedDeviceNames().size()>0)
  ListaKetai=new KetaiList(this, BlueTooth.getPairedDeviceNames());
}
else
AlertaConectado();
}


void ShowObjsTab1()
{
  Boton1.Show();
  Boton2.Show();
  AreaDeTexto.show();
}
void HideObjsTab1()
{
  Boton1.Hide();
  Boton2.Hide();
  AreaDeTexto.hide();
}

void ShowObjsTab2(){
  Dibujo.Show();
}
void HideObjsTab2(){
  Dibujo.Hide();
}

void AlertaConectado()AS,,b
{
  KetaiAlertDialog.popup(this, "Ya hay conexion Con", ListaKetai.getSelection());
```

Lo siguiente que habría que hacer sería configurar correctamente la pestaña Bluetooth para poder hacer uso de dicha herramienta de comunicación.

Bluetooth

```
void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  BlueTooth= new KetaiBluetooth(this);
}

void onActivityResult(int requestCode, int resultCode, Intent data)
{
  BlueTooth.onActivityResult(requestCode, resultCode, data);
}

void onKetaiListSelection(KetaiList Klist)
{
  Seleccion=Klist.getSelection();
  BlueTooth.connectToDeviceByName(Seleccion);
  if(Seleccion!="")
  Conectado=true;
  Klist=null;
}

void onBluetoothDataEvent(String who, byte[] data)
{
  if(Pestanas.Tab2)
  {
    if(String.valueOf(data[0])!="" && String.valueOf(data[1])!="")
    {
      CirculoXi=int(data[0]);
      CirculoYi=int(data[1]);
      DatosBluetooth=true;
    }
  }
}
```

Para que todas las pestañas anteriores funcionen correctamente y de forma coordinada es necesario añadir una pestaña, la cual denominaremos Tabs2, y que se encarga de realizar las configuraciones básicas del programa.

Tabs2

```
public class Tabs2
{
int TamanoTextoTab1, TamanoTextoTab2;
color ColorTab1Activa, ColorTab1Inactiva, ColorTab1Pressed;
color ColorTab2Activa, ColorTab2Inactiva, ColorTab2Pressed;
color ColorTextoActivoTab1, ColorTextoInactivoTab1;
color ColorTextoActivoTab2, ColorTextoInactivoTab2;
String TextoTab1, TextoTab2;
boolean Tab1, Tab2, Tab1Dragged, Tab2Dragged;

Tabs2(String Tt1, int Ttt1, color Ctat1, color Ctit1, color Cta1, color Cti1, color Ctp1,
      String Tt2, int Ttt2, color Ctat2, color Ctit2, color Cta2, color Cti2, color Ctp2)
      {
        HideObjsTab2();
        ShowObjsTab1();
        smooth();
        noStroke();
        TextoTab1=Tt1;
        TamanoTextoTab1=Ttt1;
        ColorTextoActivoTab1=Ctat1;
        ColorTextoInactivoTab1=Ctit1;
        ColorTab1Activa=Cta1;
        ColorTab1Inactiva=Cti1;
        ColorTab1Pressed=Ctp1;
        TextoTab2=Tt2;
        TamanoTextoTab2=Ttt2;
        ColorTextoActivoTab2=Ctat2;
        ColorTextoInactivoTab2=Ctit2;
        ColorTab2Activa=Cta2;
        ColorTab2Inactiva=Cti2;
        ColorTab2Pressed=Ctp2;
        Tab1=true;
        Tab2=false;
        Tab1Dragged= false;
        Tab2Dragged= false;
        DibujarTabs();

      }

      void Pressed()
      {
        if(mouseX>0 && mouseX<width/2 &&
           mouseY>0 && mouseY<2*uY)
           {
             Tab1=true;
             Tab2=false;
             fill(ColorTab1Pressed);
             rect(0,0,width/2,2*uY);
             textSize(uY);
             fill(ColorTextoActivoTab1);
             textAlign(CENTER,CENTER);
             text(TextoTab1, int(4.5*uX),uY);
           }
           if(mouseX>width/2 && mouseX<width &&
              mouseY>0 && mouseY<2*uY)
              {
             Tab1=false;
             Tab2=true;
             fill(ColorTab2Pressed);
             rect(width/2,0,width/2,2*uY);
             textSize(uY);
             fill(ColorTextoActivoTab2);
             textAlign(CENTER,CENTER);
             text(TextoTab2, int(13.5*uX),uY);
              }

      }
```

```
void Dragged()
{
  if(mouseX>0 && mouseX<width/2 &&
     mouseY>0 && mouseY<2*uY)
     Tab1Dragged=true;

  if(mouseX>width/2 && mouseX<width &&
     mouseY>0 && mouseY<2*uY)
     Tab2Dragged=true;

}


void Released(){
  if(mouseX>0 && mouseX<width/2 &&
     mouseY>0 && mouseY<2*uY){
       HideObjsTab2();
       fill(g.backgroundColor);
       rect(0,2*uY+1,width,30*uY);
       ShowObjsTab1();
       DibujarTabs();
     }
     if(mouseX>width/2 && mouseX<width &&
     mouseY>0 && mouseY<2*uY){
       HideObjsTab1();
       fill(g.backgroundColor);
       rect(0,2*uY+1,width,30*uY);
       ShowObjsTab2();
       DibujarTabs();
     }
  if(Tab1Dragged){
    Tab1Dragged=false;
    HideObjsTab2();
    fill(g.backgroundColor);
    rect(0,2*uY+1,width,30*uY);
    ShowObjsTab1();
    DibujarTabs();
  }
  if(Tab2Dragged){
   Tab2Dragged=false;
    HideObjsTab1();
    fill(g.backgroundColor);
    rect(0,2*uY+1,width,30*uY);
    ShowObjsTab2();
    DibujarTabs();
  }
}


 void DibujarTabs()
 {
   if(Tab1)
   {
     noStroke();
     fill(ColorTab1Activa);
     rect(0,0,width/2,2*uY);
     fill(ColorTab2Inactiva);
     rect(width/2,0,width/2,2*uY);
     textSize(uY);
     fill(ColorTextoActivoTab1);
     textAlign(CENTER,CENTER);
     text(TextoTab1, int(4.5*uX),uY);
     fill(ColorTextoInactivoTab2);
     textAlign(CENTER,CENTER);
     text(TextoTab2, int(13.5*uX),uY);
   }

   if(Tab2)
   {
     noStroke();
     fill(ColorTab1Inactiva);
     rect(0,0,width/2,2*uY);
     fill(ColorTab2Activa);
     rect(width/2,0,width/2,2*uY);
     textSize(uY);
     fill(ColorTextoActivoTab2);
     textAlign(CENTER,CENTER);
     text(TextoTab2, int(13.5*uX),uY);
     fill(ColorTextoInactivoTab1);
     textAlign(CENTER,CENTER);
     text(TextoTab1, int(4.5*uX),uY);
   }

 }
}
```

Una vez se tiene la pestaña Tabs2 correctamente configurada ya tan sólo faltaría configurar la pestaña principal, que sería la que se encargaría de interactuar con el joystick y realizar el movimiento del objeto deseado.

Joystick | Boton | Tabs2

```
import ketai.ui.*;
import controlP5.*;
import ketai.net.*;
import android.os.Bundle;
import ketai.net.bluetooth.*;
import android.content.Intent;

int uX,uY,Contador;
int CirculoXi,CirculoYi;
float x1,y1;
float CirculoXf1,CirculoYf1;
float CirculoXf2,CirculoYf2;
byte[] Dato;
Tabs2 Pestanas;
String Seleccion;
boolean Conectado, Seguir, DatosBluetooth;
Println Consola;
Textarea AreaDeTexto;
KetaiList ListaKetai;
ControlP5 Cp5;
BotonRedondo Boton1, Boton2;
AreaDeDibujo Dibujo;
KetaiBluetooth BlueTooth;

void setup()
{
  size(1900,1050);
  uX=width/32;
  uY=height/18;
  Seleccion="";
  Seguir=true;
  Conectado=false;
  Cp5=new ControlP5(this);
  BlueTooth.start();
  background(#9E9E9E);
  CirculoXf1=width/2;
  CirculoYf1=height/2;
  DatosBluetooth=false;
  ellipseMode(CENTER);
  AreaDeTexto = Cp5.addTextarea("Txt").setPosition(3*uX, int(2.5*uY))
                                      .setSize(26*uX, int(10.5*uY))
                                      .setFont(createFont("",uY))
                                      .setLineHeight(2*uY)
                                      .setColor(color(#00BFA5))
                                      .setColorBackground(color(#616161))
                                      .setColorForeground(color(#C51162));
  Consola = Cp5.addConsole(AreaDeTexto);

  Boton1=new BotonRedondo(5*uX, int(15.5*uY), 4*uX, color(#0091EA), color(#40C4FF),
                          "Buscar", int(0.8*uY),color(#004D40));
  Boton2=new BotonRedondo(27*uX, int(15.5*uY), 4*uX, color(#0091EA), color(#40C4FF),
                          "Conectar", int(0.8*uY),color(#004D40));

  Dibujo = new AreaDeDibujo(uX, 2.5*uY, 30*uX, 15*uY, color(#CFD8DC));

  Pestanas = new Tabs2("Configuracion", uY, color(#FFD600), color(#FFFDE7),
                       color(#2962FF),color(#BBDEFB),color(#82B1FF),
                       "Joystick", uY, color(#FFD600), color(#FFFDE7),
                       color(#2962FF),color(#BBDEFB),color(#82B1FF));

}

void draw()
{
  if(DatosBluetooth)
  {
   CirculoXf2=map(CirculoXi,0,255,Dibujo.Xi+uX,Dibujo.Xf-uX);
   CirculoYf2=map(CirculoYi,0,255,Dibujo.Yi+uY,Dibujo.Yf-uY);
   noStroke();
   fill(#CFD8DC);
   ellipse(CirculoXf1, CirculoYf1, 2*uX+1, 2*uY+1);
   fill(#F44336);
   ellipse(CirculoXf2, CirculoYf2, 2*uX, 2*uY);
   CirculoXf1=CirculoXf2;
   CirculoYf1=CirculoYf2;
   DatosBluetooth=false;
  }
  }
```

Con esta última pestaña el código del programa quedaría completo y ya se podría ejecutar. Lo que se obtendría si se ejecuta dicho código en una terminal Android sería, por un lado, una primera pantalla dónde se tendría que elegir a que terminar Bluetooth se quiere conectar. Para interactuar con dicha pantalla se deberá pulsar el botón izquierdo de la pantalla para visualizar un desplegable con todas las opciones al alcance. Con el botón derecho se desplazaría por las opciones y al pulsar de nuevo el botón izquierdo se seleccionaría la opción elegida.

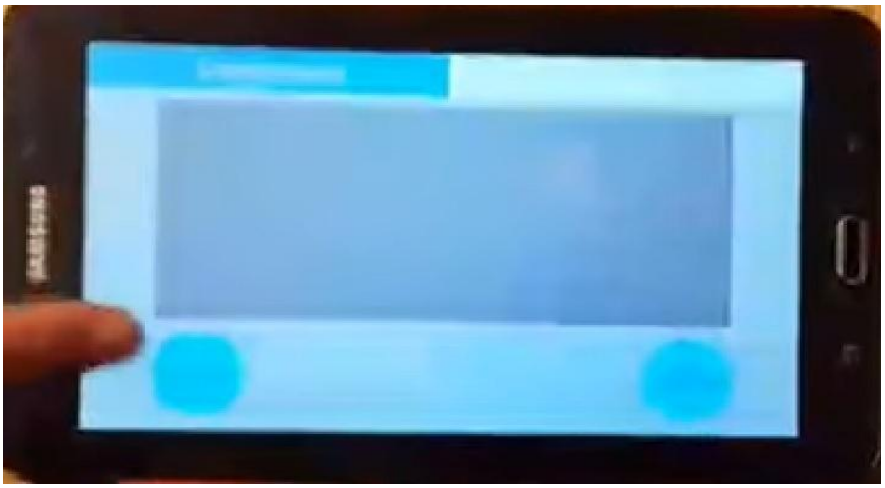


Figura 37. Demostración processing 1

Una vez se elija una opción la pantalla cambiaría a la segunda pantalla del programa, en la cual se mostraría un círculo rojo sobre un fondo blanco, el cual se movería sobre dicho fondo siguiendo el movimiento que describiese el joystick.

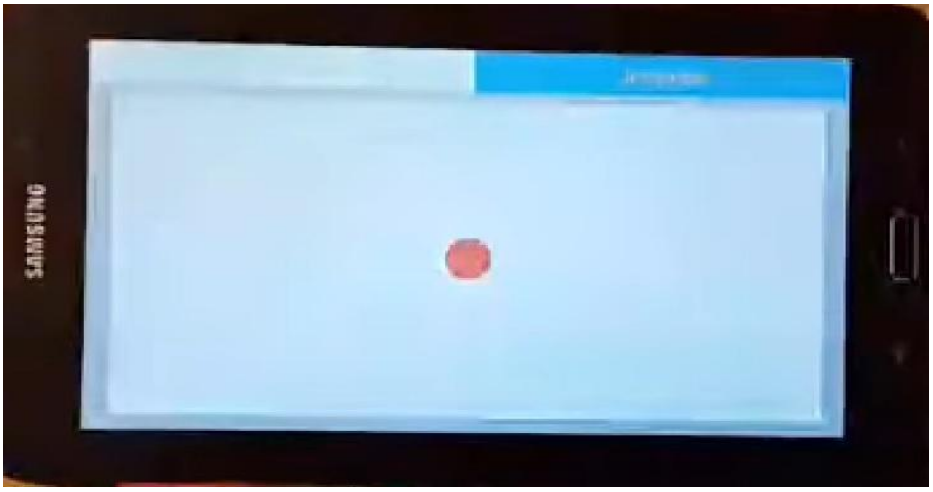


Figura 38.Demostracion processing 2

Lamentablemente, como se comentó anteriormente, al tener un problema con una de las librerías para el uso del bluetooth en la terminal Android utilizada, esta opción no ha sido finalmente viable para el proyecto, que se terminó realizando con otro entorno de desarrollo totalmente diferente. A pesar de ello se quiso dejar constancia del trabajo realizado, por si algún día pueda ser de ayuda a alguien.

Ahora se pasará a enseñar el desarrollo de la opción escogida para el proyecto y se explicará paso a paso como se ha realizado la programación por bloques de esta.

Al empezar, nada más abrir la aplicación, se encontrará la pantalla de inicio, que es la siguiente.

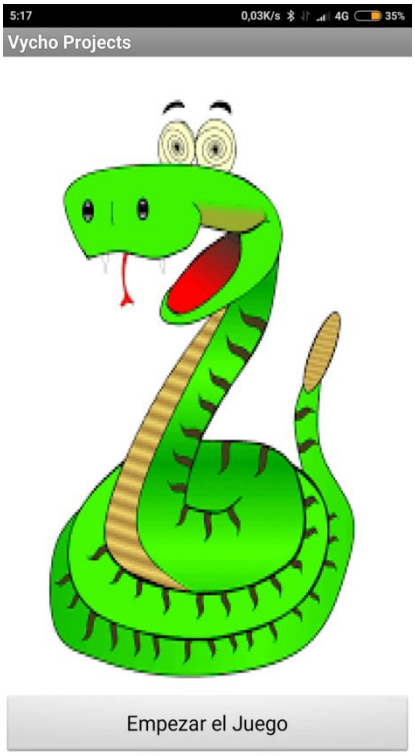
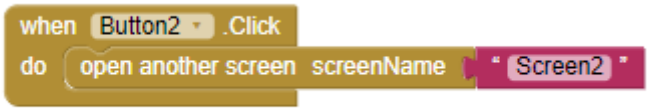


Figura 39 Pantalla Inicial de la aplicación,

Para diseñar dicha pantalla simplemente se ha añadido un lienzo (Canvas en inglés) y se ha añadido a dicho lienzo la imagen importada desde la computadora. Además se ha colocado un botón abajo que contiene el texto (Caption en inglés) <<Empezar el Juego>> que al pulsarlo abre una segunda ventana que sustituye a la primera. Dicho botón se ha programado de la siguiente manera.



Cuando se abre esta segunda pantalla lo primero que aparece es un mensaje que informa que el Bluetooth debe estar conectado. Al aceptar dicho mensaje desaparece, quedando la aplicación en la segunda pantalla, dónde solamente se puede o pulsar el botón de Reset, el cual llevaría la aplicación a la primera pantalla, o pulsar el botón de Seleccionar Dispositivo, el cual desplegaría una lista de opciones a las que conectar el dispositivo Android mediante bluetooth.

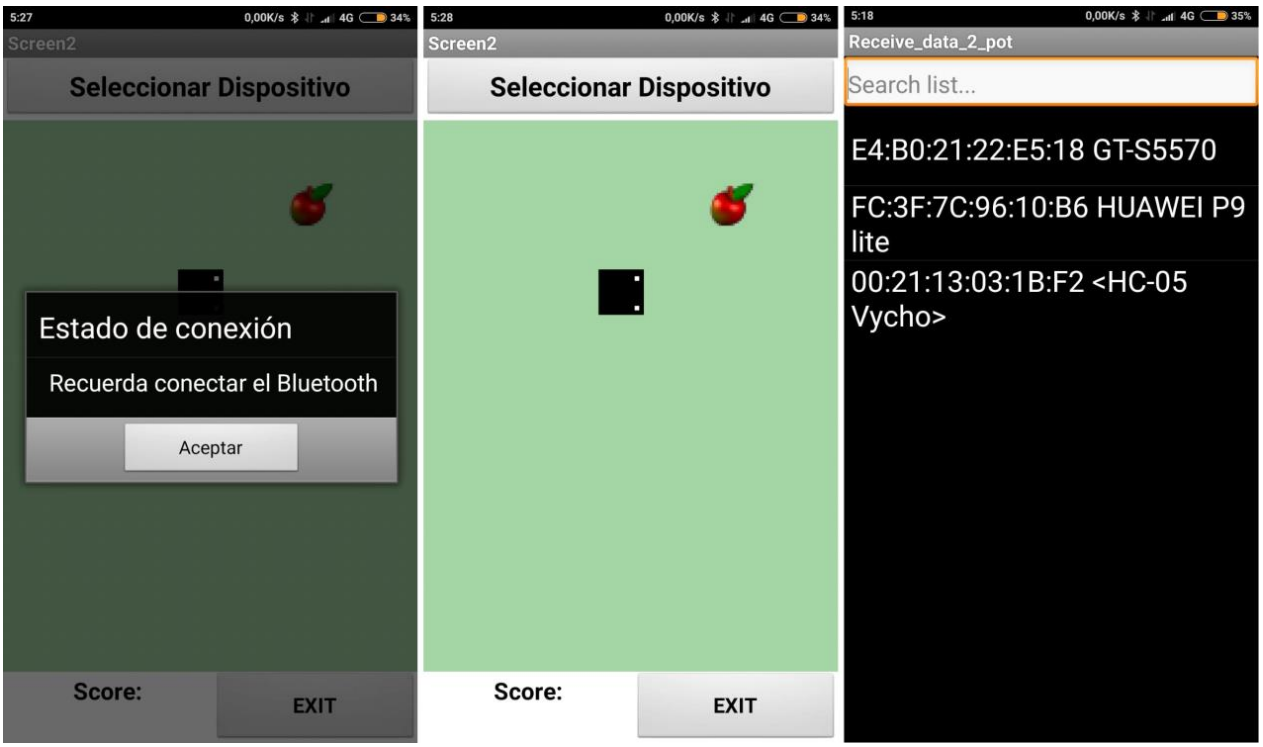
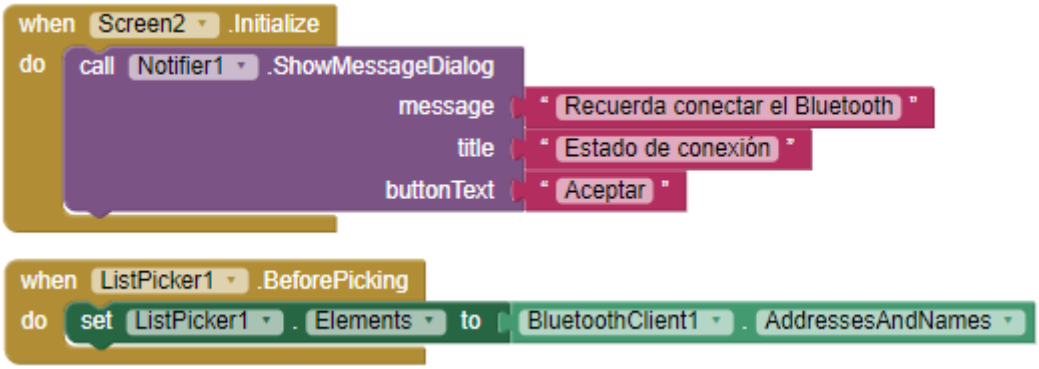


Figura 40 Interfaz de conexión Bluetooth.

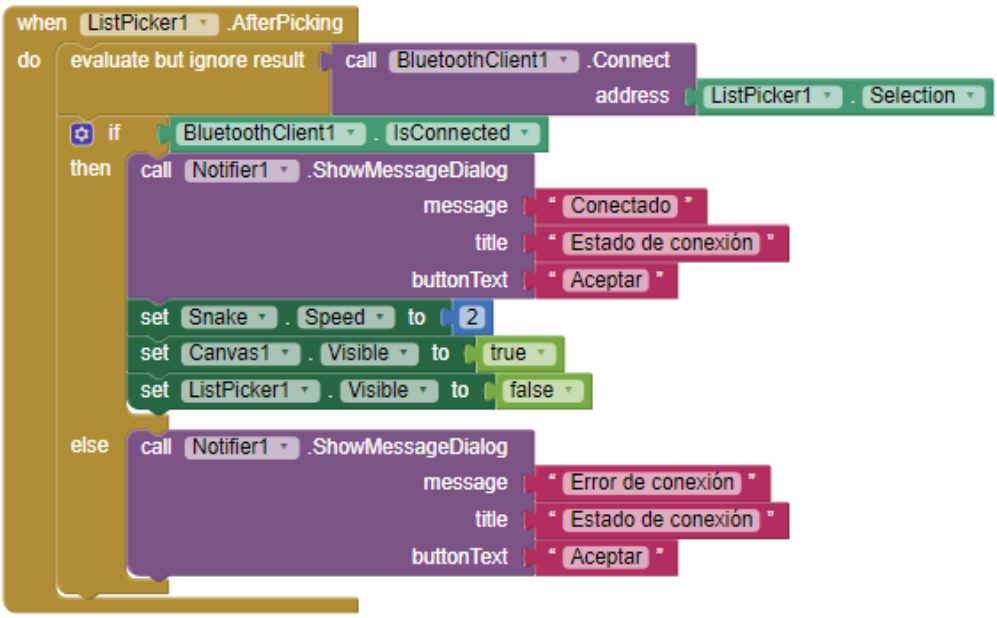
La programación de estas interacciones ha sido la siguiente.



Una vez se pulse en alguna de las opciones para conectar mediante bluetooth, hay dos opciones, la primera, que todo funcione correctamente, en cuyo caso aparecerá un mensaje informando de que la terminal está conectada. Cuando se acepte dicho mensaje aparecerá un cero en el contador de abajo (Score) y el juego comenzará. La segunda opción es que por algún motivo la conexión falle, en cuyo caso aparecerá un mensaje informando de que ha habido un error de conexión, y el juego no empezara, teniendo que repetirse la acción anterior para poder dar comienzo al juego.

Figura 41 Resultado de la conexión.

La programación de dicha función ha sido la siguiente.



Una vez el juego haya comenzado este consistirá en mover mediante el joystick a una pequeña serpiente cuadrada que avanzará en búsqueda de una manzana que aparecerá en cualquier recuadro aleatorio del campo de juego. Cuando la serpiente consiga comerse una de estas frutas aumentará en uno el valor del contador (Score) aumentando así la puntuación y la velocidad a la que se mueve. El juego termina cuando la serpiente colisiona con alguno de los bordes que delimitan el campo de juego, es decir, los márgenes, apareciendo entonces un mensaje que indicará fin del juego (Game Over), el cual al ser aceptado mostrará un mensaje con la puntuación final alcanzada. Además, el mando consta de un botón, que, en caso de ser pulsado, reiniciará el juego, llevando el contador a cero y la velocidad de la serpiente a la velocidad inicial. Por último, el diseño cuenta con un botón llamado Exit, que en caso de ser pulsado indicará con un mensaje que se ha abandonado el juego (Game Out) y que lanzará la pantalla inicial 1.

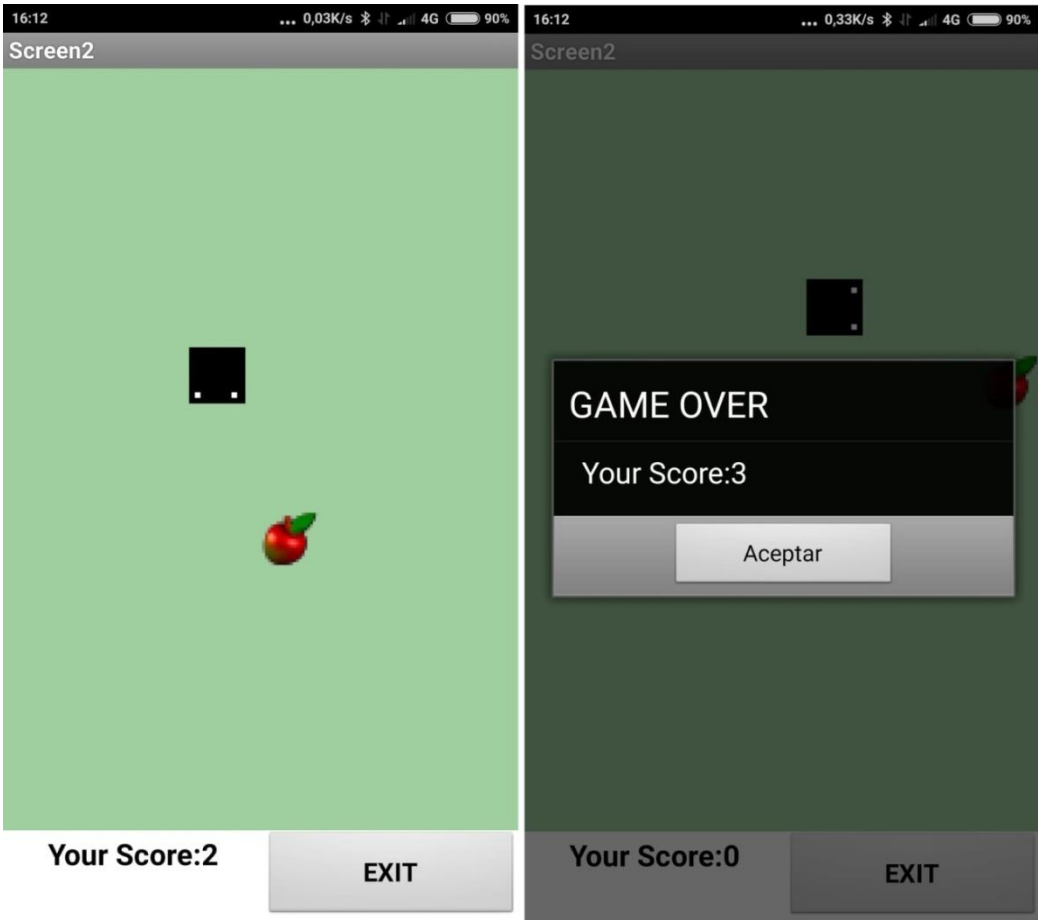
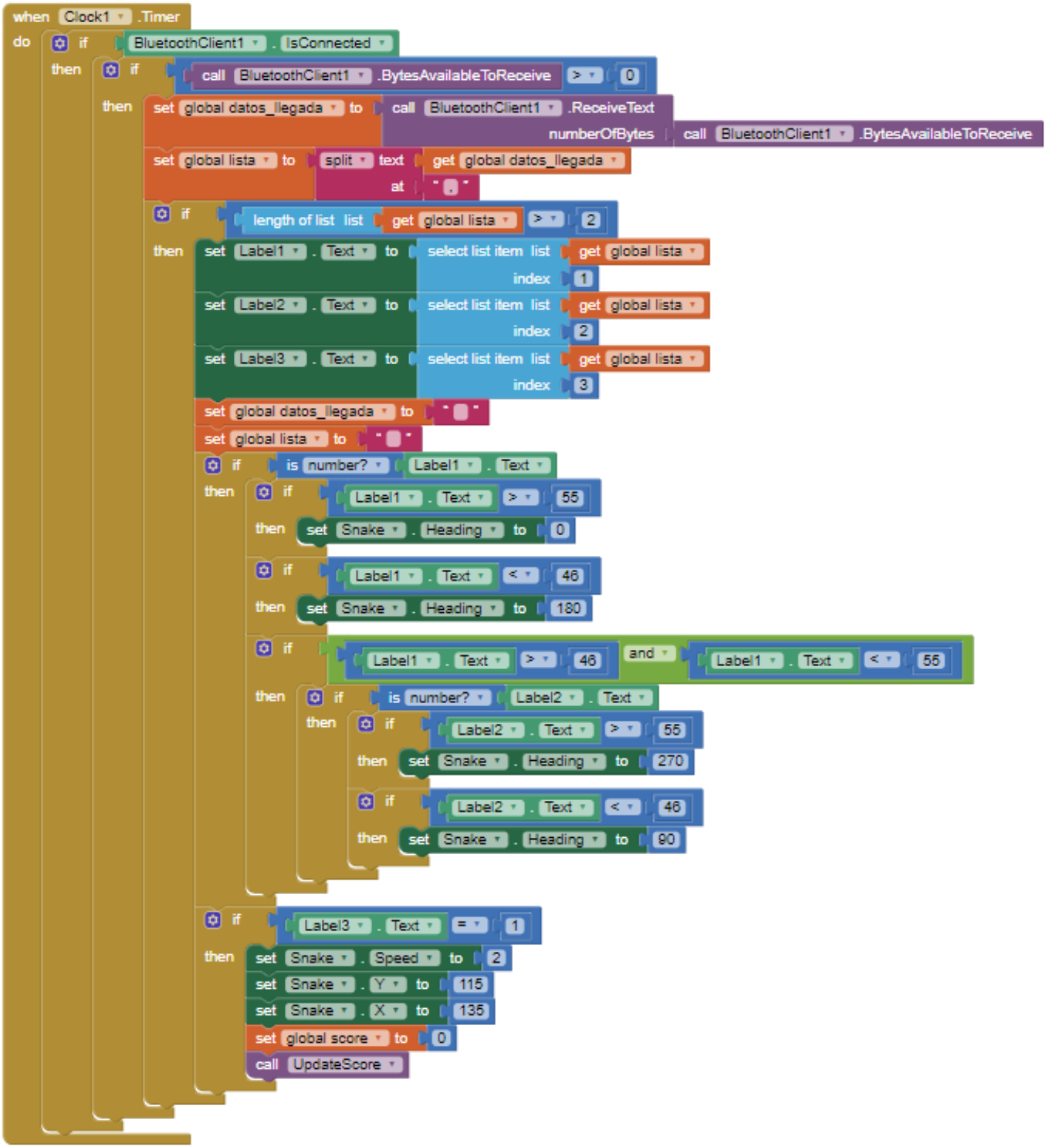
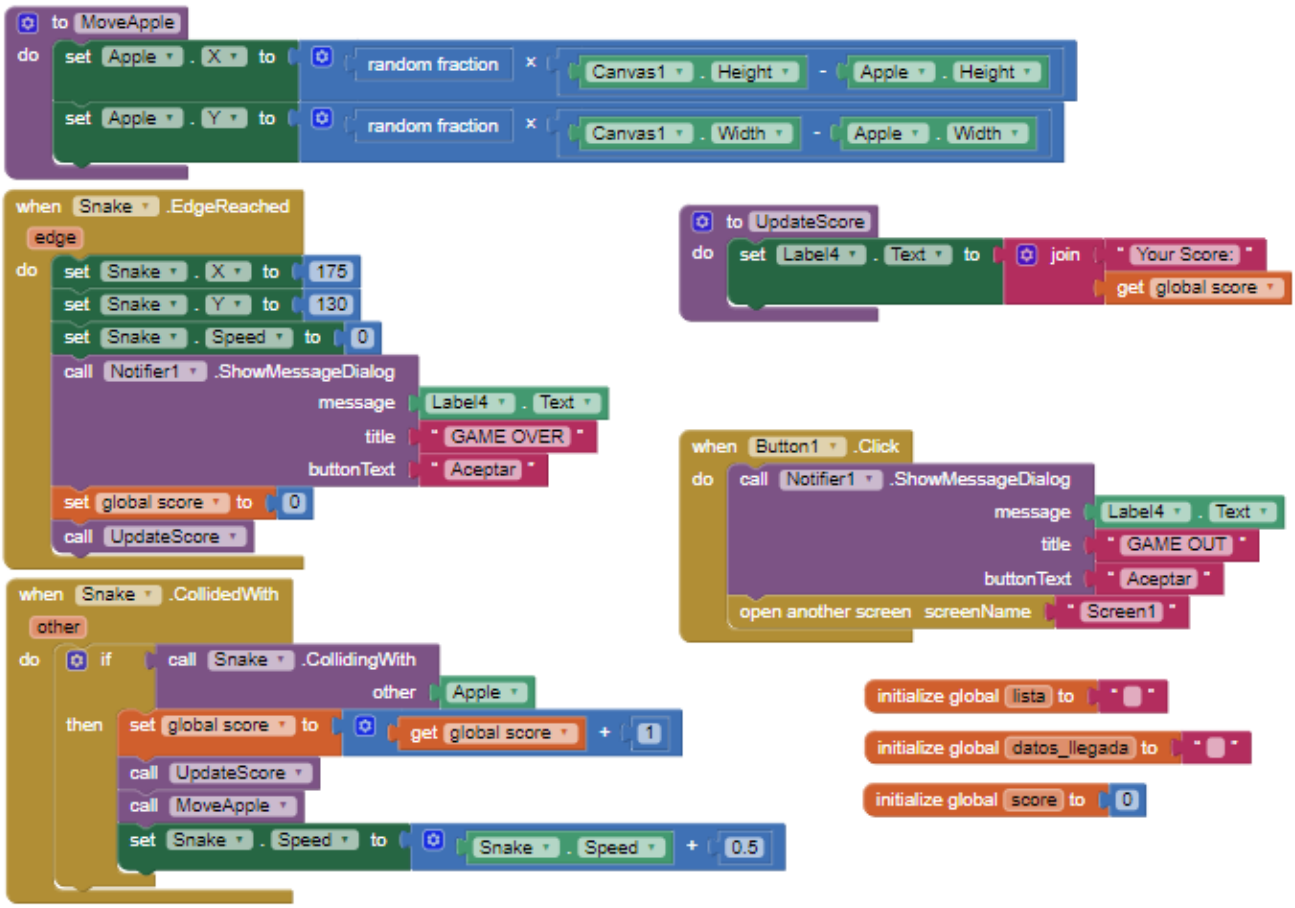


Figura 42 Interfaz visual del videojuego.

El código principal del juego se desarrolla todo el tiempo haciendo uso de un timer, cuya programación es la siguiente.



Además, para completar la programación de la aplicación se han utilizado algunas variables y varias funciones, que son las siguientes.



4. Comprobación de correcto funcionamiento del proyecto.

Para comprobar el correcto funcionamiento del proyecto se han efectuado pruebas en cuatro dispositivos Android diferentes, obteniendo unos resultados satisfactorios en todos ellos. A pesar de los buenos resultados, se ha podido comprobar que hay un pequeño error. Dicho error consiste en que algunas veces, y sin seguir ningún patrón ni ninguna secuencia, el joystick altera durante unas décimas de segundo el valor del eje X, haciendo así que la serpiente cambie de trayectoria durante una pequeña porción de tiempo. Después de efectuar varias pruebas para tratar de solucionar dicho error, e ir analizado todo el sistema en búsqueda del foco del error, se ha alcanzado la conclusión de que dicho error procede del montaje electrónico del joystick, ya que el error no procedía ni de la aplicación diseñada, ni del módulo bluetooth. Esto último se ha comprobado analizando los valores que daba el joystick en la pantalla serial del Arduino Uno, dando valores reiterados de 51-52 en el eje X cuando el joystick de encontraba colocado en el medio, y apareciendo un valor de 0 algunas veces sin previo aviso ni motivo alguno.

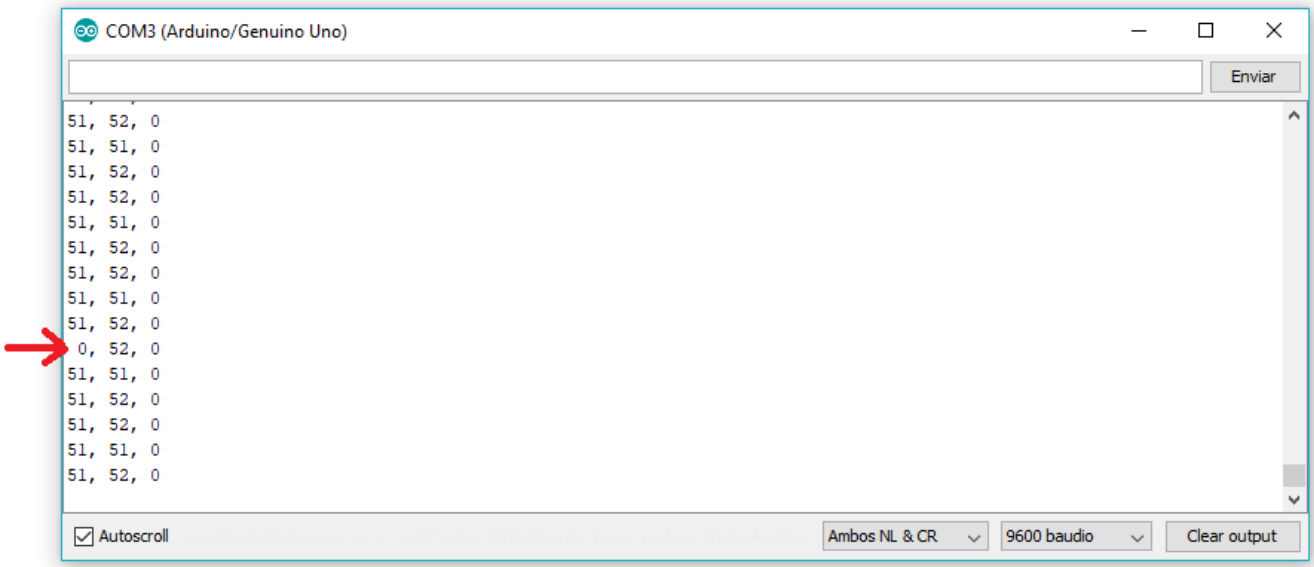


Figura 43 Puerto serie del Arduino

Finalmente dicho error no pudo ser solucionado, por suerte es poco importante para el desarrollo del proyecto y finalmente se puede dar validez a los resultados obtenidos.

5. Conclusiones.

Durante la realización del proyecto se han podido alcanzar una serie de conclusiones muy reseñables.

La primera de ellas es que, por muchos impedimentos que puedan surgir a la hora de elaborar un proyecto (Sin ir más lejos, la imposibilidad de continuar el proyecto utilizando el entorno de desarrollo APDE Processing, con el cual se tardó cerca de un mes para familiarizarse con su uso, por motivos ajenos a los conocimientos, sino a factores externos como la marca del smartphone y/o su versión) siempre se puede realizar, ya sea cambiando en enfoque que se le ha podido dar en un principio, o modificando los medios elegidos para su realización.

Otra de las conclusiones extraídas de este proyecto ha sido la importancia de tener una comunidad de programadores que comparten conocimientos de forma totalmente altruista, ya que, por muchas dudas que surgieran durante la programación, siempre había algún lugar recóndito de internet en el que alguien explicaba claramente donde estaba el error y como solucionarlo. Si antes se ha mencionado que se tardó cerca de un mes en aprender a programar en processing de manera correcta, se puede afirmar sin error que sin dicha comunidad de programadores el tiempo habría sido probablemente más del doble.

La última conclusión extraída es que, por suerte, los componentes electrónicos básicos que se utilizan prácticamente en cualquier proyecto están al alcance del bolsillo de prácticamente todo el mundo, pudiendo así potenciar jóvenes mentes que, aunque carezcan de una fuente sustancial de riqueza, pueden aportar mucho al mundo de la electrónica con sus conocimientos, ideas, y ganas de aprender.

6. Bibliografía.

- Luís Llamas (Sin fecha) - Informe sobre alimentación de baterías. Encontrado en la siguiente URL: https://www.luisllamas.es/alimentar-Arduino-baterias/

- Descripción básica de programación en Arduino. Encontrado en la siguiente URL: http://panamahitek.com/el-setup-y-el-loop-en-Arduino/

- Informe sobre la historia de los microprocesadores. Encontrando en la siguiente URL: http://www.maestrosdelweb.com/historia-de-los-microprocesadores/

- Informe sobre la evolución de los microprocesadores en los últimos 40 años. Encontrado en la siguiente URL: https://www.muycomputer.com/2011/11/15/evolucion-de-los-microprocesadores-40-anos-de-historia-infografia/

- Explicación básica del funcionamiento de los micropagos. Encontrado en la siguiente URL: https://www.gestion.org/el-sistema-de-micropagos/

- Introducción básica a Processing. Encontrado en la siguiente URL: http://www.programacionyrobotica.com/introduccion-processing/

- Durante todo el proyecto se han consultado tanto algunas definiciones como algunas fechas de la siguiente URL: https://www.wikipedia.org/

- Informe sobre la historia de Java. Encontrado en la siguiente URL: http://www.tuprogramacion.com/programacion/historia-de-java/

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

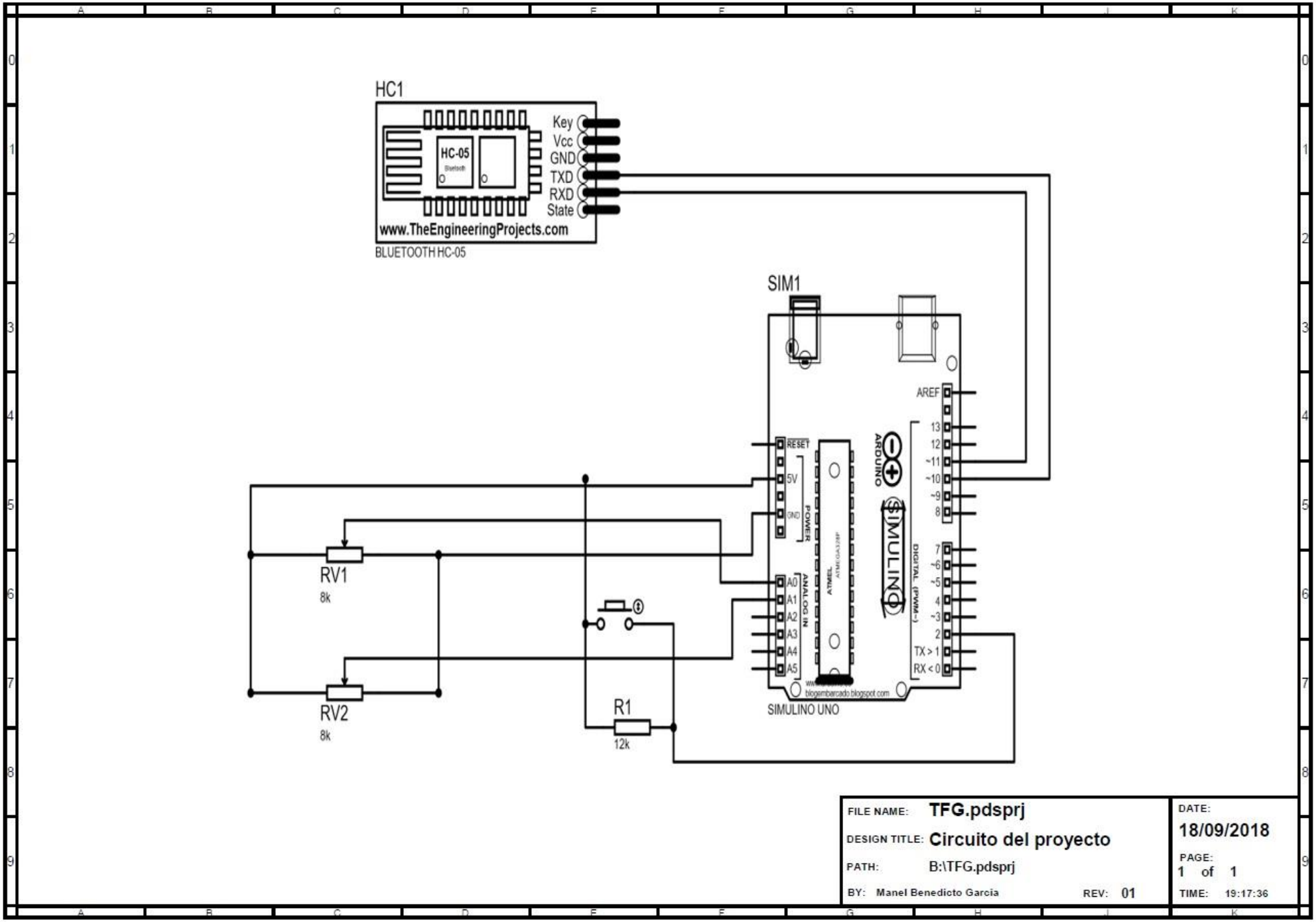
DESARROLLO DE UNA INTERFAZ HOMBRE MÁQUINA BASADA EN JOYSTIC PARA EL CONTROL DE APLICACIONES ANDROID

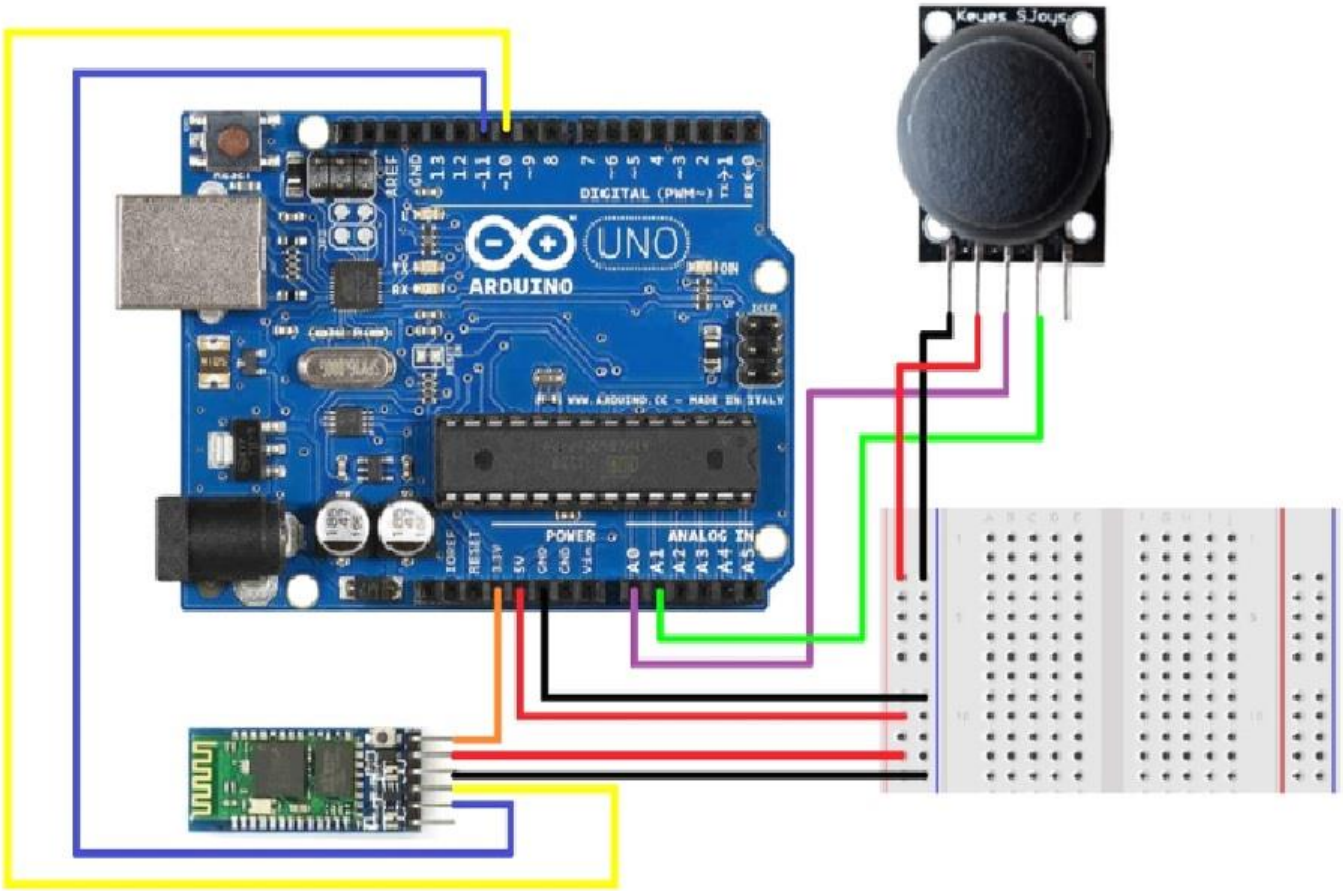
Trabajo Fin de Grado en Ingeniería Electrónica Industrial y Automática

Documento 2: Planos

Autor: Manuel Benedicto García

Tutor: Carlos Pascual Domínguez Montagud


HC1
Key
Vcc
GND
TXD
RXD
State
HC-05
www.TheEngineeringProjects.com
BLUETOOTH HC-05
SIM1
AREF
RESET
5V
GND
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
ATMEL
ARDUINO
SIMULINO
DIGITAL (PWM~)
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX > 1
RX < 0
blogembarcado.blogspot.com
SIMULINO UNO
RV1
8k
RV2
8k
R1
12k
FILE NAME: TFG.pdsprj
DESIGN TITLE: Circuito del proyecto
PATH: B:\TFG.pdsprj
BY: Manel Benedicto Garcia
REV: 01
DATE: 18/09/2018
PAGE: 1 of 1
TIME: 19:17:36



FILE NAME:	TFG.pdsprj	DATE:	18/09/2018
DESIGN TITLE:	Circuito del proyecto	PAGE:	1 of 1
PATH:	B:\TFG.pdsprj	TIME:	19:17:36
BY:	Manel Benedicto Garcia	REV:	01

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO DE UNA INTERFAZ HOMBRE MÁQUINA BASADA EN JOYSTIC PARA EL CONTROL DE APLICACIONES ANDROID

Documento 3: Presupuesto

Autor: Manuel Benedicto Garcia

Tutor: Carlos Pascual Domínguez Montagud

Trabajo Fin de Grado en Ingeniería Electrónica Industrial y Automática

Análisis porcentual de unitarios

Código	Designación	Importe total	% PEM
AUNO	Arduino UNO	21,00	46,71
PWB5V	Power Bank 5V/3.5A	9,22	20,51
MBT	Modulo Bluethooth HC-05	3,29	7,32
MOE	Especialista en Electrónica	2,50	5,56
CUSB	Cable de conexión USB	1,94	4,31
JT	Joystick Arduino	1,50	3,34
CAP	Caja pequeña de madera	1,46	3,25
	T o t a l	40,91	

Análisis porcentual de unidades de obra

Código	Designación	Importe total	% PEM
AUNO	Arduino UNO	21,63	51,31
PWB5V	Power Bank 5V/3.5A	9,50	22,54
MBT	Modulo Bluethooth HC-05	3,39	8,04
MOE	Especialista en Electrónica	2,58	6,12
CUSB	Cable de conexión USB	2,00	4,74
JT	Joystick Arduino	1,55	3,68
CAP	Caja pequeña de madera	1,50	3,56
	T o t a l	42,15	

Análisis porcentual de capítulos

Código	Designación	Importe total	% PEM
C01	**componentes**	**42,38**	**93,88**
C02	**montaje**	**2,58**	**6,12**
	T o t a l	42,15	

Análisis por naturaleza

PRESUPUESTO PROYECTO

Código	Ud	Resumen	Cantidad	Materiales	%	Maquinaria	%	Mano de obra	%	Otros	%	Importe
Capítulo C01 componentes												
PWB5V	UD	Power Bank 5V/3.5A	1,000							9,50	22,5	9,50
AUNO	UD	Arduino UNO	1,000							21,63	51,3	21,63
CUSB	UD	Cable de conexión USB	1,000							2,00	4,7	2,00
MBT	UD	Modulo Bluethooth HC-05	1,000							3,39	8,0	3,39
JT	UD	Joystick Arduino	1,000							1,55	3,7	1,55
CAP	UD	Carcasa de mando	1,000							1,50	3,6	1,50
		Total capítulo C01								**39,57**	93,9	**39,57**
Capítulo C02 montaje												
MOE	h	Especialista en Electrónica	0,500							2,58	6,1	2,58
		Total capítulo C02								**2,58**	6,1	**2,58**
		Total obra								**42,15**	100,0	**42,15**