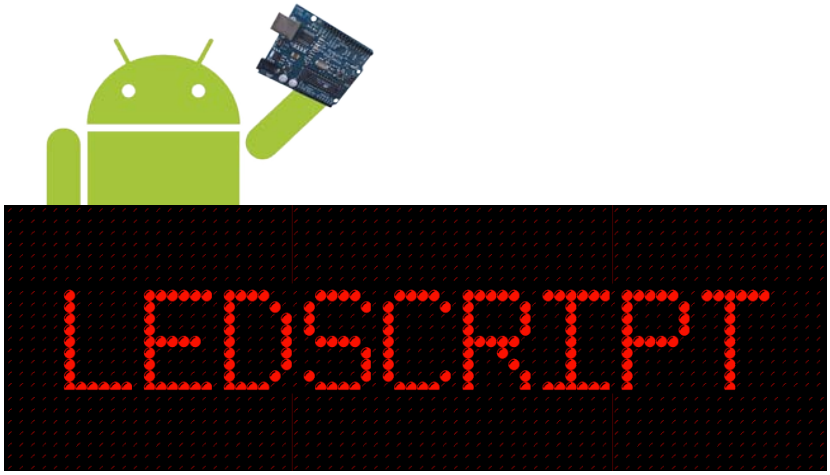
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escuela Técnica Superior de Ingeniería Informática

Universidad Politécnica de Valencia



Proyecto Final de Carrera

Ingeniería Técnica Informática de Sistemas

Autor: Iker Orte Ruiz

Director: Ángel Rodas Jordá

Septiembre de 2013

Resumen

El proyecto LEDSCRIPT se creó con la idea de diseñar un hardware que pudiera ser controlado mediante un dispositivo móvil que supliera unas necesidades concretas para facilitar y mejorar la funcionalidad de nuestro hardware.

Se valoraron diversas ideas, pero finalmente recibimos una propuesta desde la facultad de Bellas Artes en el que nos propusieron realizar un hardware que realizara una serie de funcionalidades que pudieran aplicar a un tipo de arte en concreto. No obstante debido a los cortos límites de tiempo y la falta de comunicación, así como la limitación por parte del hardware, este proyecto no pudo ser realizado.

A través de esta idea y adaptándolo a unas necesidades más genéricas surgió la idea final de nuestro proyecto LEDSCRIPT. Para llegar a nuestra meta de realizar nuestro proyecto y debido a que no se disponía de ninguna experiencia en la producción de hardware, se opto por realizar una fase previa de prototipado que aportaría la experiencia necesaria para la construcción del hardware final y facilitar la posterior codificación del software de la que también se realizaría una secuencia de demostración.

Una vez diseñado el hardware se paso a la producción del prototipo con el que se observaron las posibles deficiencias y se solventaron. Para más adelante facilitar la producción de del hardware definitivo. Por todo ello la producción del hardware definitivo fue relativamente rápido pues ya se poseía cierta experiencia en la construcción.

No obstante debido a la gran amplitud del proyecto y a la multitud de ámbitos que abarca no ha sido posible completar totalmente la consecución del mismo, que en un futuro próximo se concluirá.

Tabla de contenidos

1. Introducción

1.1 Motivación del proyecto

Hoy en día el mundo de la informática se ha diversificado enormemente englobando todos los aspectos de la vida cotidiana, muchos son los factores que han beneficiado el auge de la informática en detrimento de otras áreas de la ciencia, pero especialmente la liberalización del sector y el favorecimiento de la creación y desarrollo de nuevos productos mediante el uso de plataformas libres, tanto de software como de hardware. En este contexto en el Proyecto LEDSCRIPT tiene como mayor objetivo el uso, la difusión y la promoción de dichas plataformas para dar a conocer los amplios usos de los que nos podemos beneficiar en nuestro día a día.

Por otra parte dentro del mundo de la informática debido a su gran diversidad hay numerosas tendencias, como pueden ser los sistemas CLOUD, el posicionamiento SEM/SEO, el diseño WEB 2.0, “el internet de las cosas” , el desarrollo de tecnologías móviles o el diseño de pequeños instrumentos para mejorar nuestra calidad de vida.

1.2 Objetivos del proyecto

LEDSCRIPT desea ser un instrumento para facilitar y promover el uso de tecnologías libres, para ello se hace uso tanto de hardware como de software libre. En el caso de nuestro proyecto hacemos uso de la plataforma Android como software libre y de la plataforma Arduino como hardware libre. La unión de estas dos plataformas ha sido denominada como Amarino y pretende fomentar el diseño y construcción de nuevos hardware y acercar a todos los desarrolladores la oportunidad de crear algo novedoso de forma sencilla y barata.

El objetivo principal del proyecto LEDSCRIPT es realizar una aplicación para dispositivos móviles con sistema operativo libre Android que capture y un patrón y mediante el modulo bluetooth del terminal móvil hacia un receptor de bluetooth conectado al hardware libre Arduino , el cual reproducirá de forma dinámica en un panel de LEDs de diseño y creación propia. Para llevar a cabo dicho proyecto serán necesarios la adquisición de una placa Arduino UNO , un receptor de bluetooth , un dispositivo móvil con sistema operativo Android y la construcción de un panel de LEDs , como se puede apreciar en la siguiente imagen:

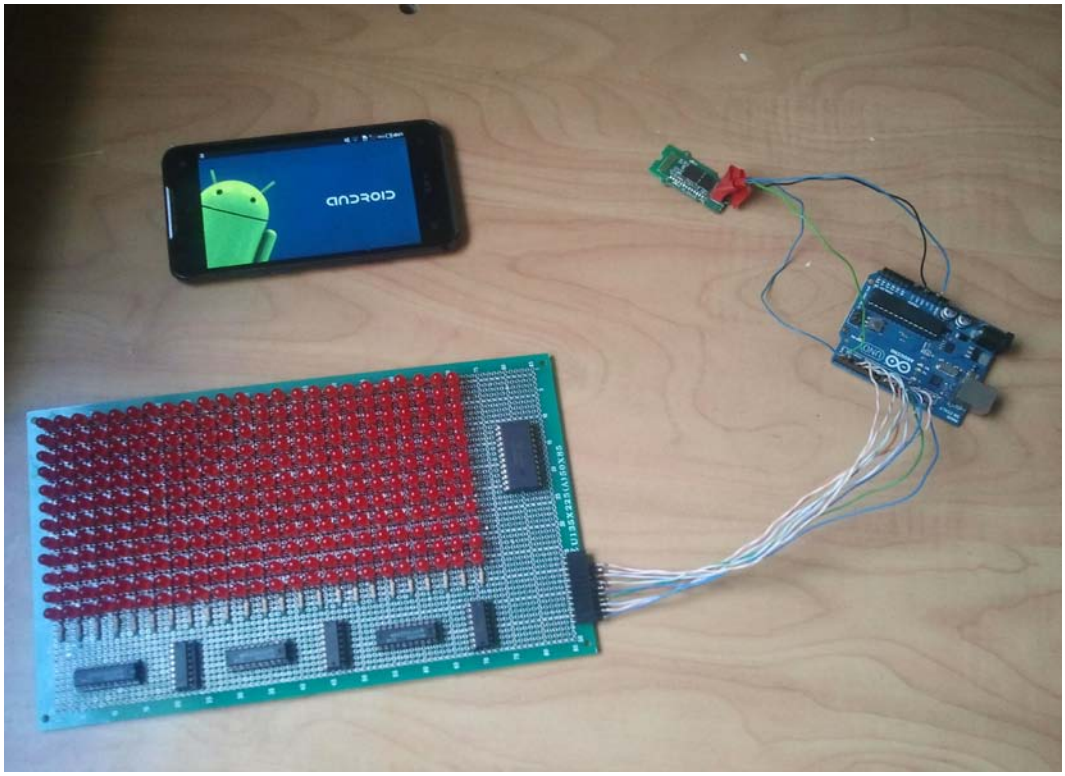


Figura 1 : Montaje LEDSCRIPT

Figura 2: símbolo Arduino

2. ARDUINO: El Opened Hardware

Arduino es una plataforma de open hardware basada en una sencilla placa y en un entorno de desarrollo que facilita la interacción con el usuario y de tal forma el desarrollo del software para esta plataforma.

Dentro de la plataforma Arduino existen múltiples dispositivos dentro de la propia plataforma que se han desarrollado con diferentes características para mejorar algunos aspectos específicos de la plataforma, como pueden ser los siguientes:

Figura 3 : Arduino Mega

Arduino MEGA

Es una placa electrónica basada en el ATmega2560. Tiene 54 pines digitales de entrada / salida, 16 entradas analógicas, 4 UARTs , un oscilador de 16MHz, una conexión USB, un conector de alimentación, una cabecera ICSP, y un botón de reinicio.

Arduino NANO

El Arduino Nano es una placa pequeña, completa, pensada para ser utilizado en placas de prototipado (Breadborads)y su electrónica esta basada en el ATmega328 . Tiene más o menos la misma funcionalidad de la Arduino Duemilanove, pero en un paquete diferente. Le falta sólo un conector de alimentación DC, y trabaja con un cable USB Mini-B en lugar de uno normal.

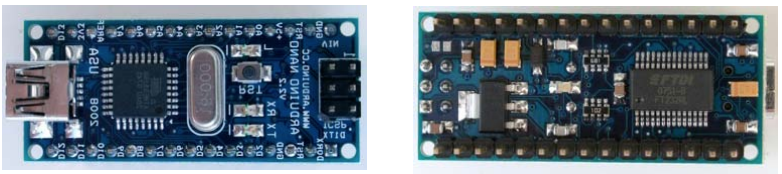


Figura 4: Arduino Nano

LilyPad Arduino

El LilyPad Arduino es una placa electronica diseñado para ropa y e-textiles. Puede ser cosida a la tela y se monta de manera similar fuentes de alimentación, sensores y actuadores con hilo conductor. La placa electrónica se basa en la ATmega168V de bajo consumo.

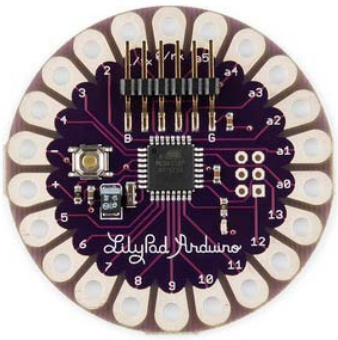


Figura 5 : Arduino Lilypad

Arduino Yún

El Arduino Yun es una placa electrónica basada en los procesadores ATMEGA32U4 y Atheros AR9331. El procesador Atheros es compatible con una distribución Linux basada en OpenWRT llamado Linino. La placa lleva incorporado un soporte para Ethernet y WiFi, un puerto USB-A, ranura para tarjeta micro-SD, 20 pines de entrada / salida digitales, un cristal oscilador de 16 MHz, una conexión micro USB, una cabecera ICSP, y 3 botones de reinicio.

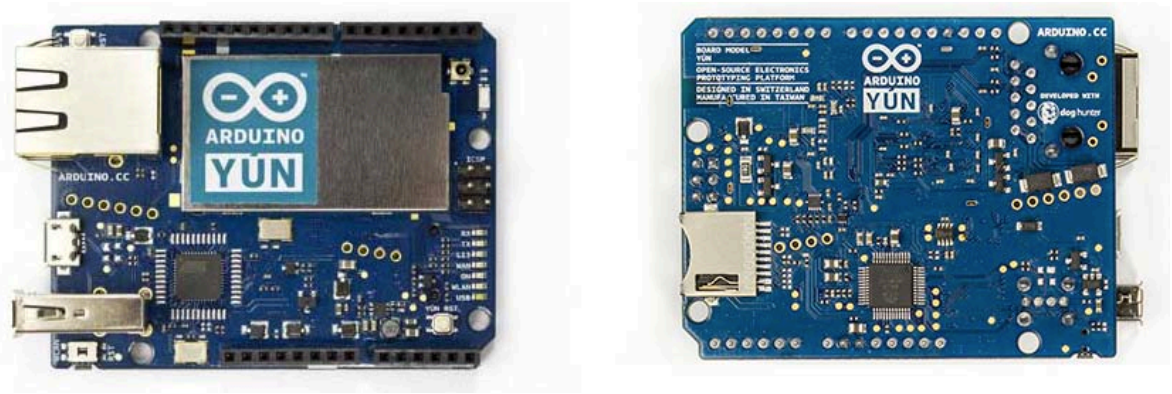


Figura 6 : Arduino YUN

Arduino UNO

También llamado Duemilanove es una placa electrónica basada en el microprocesador Atmega328 de Atmel. Tiene 14 pines digitales de entrada / salida (de las cuales 6 se puede utilizar como salidas PWM), 6 entradas analógicas, un resonador cerámico de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP, y un botón de reinicio. Contiene todo lo necesario para apoyar el micro controlador, basta con conectarlo a un ordenador con un cable USB o el poder con un adaptador AC-DC o batería para empezar.

Figura 7 : Arduino UNO

En nuestro caso hemos elegido el Arduino UNO ya que era el dispositivo de costo más bajo y sus características eras suficientes para el desarrollo de nuestro proyecto.

Sus especificaciones técnicas son las siguientes:

Microcontrolador	ATmega328
Voltaje operativo	5 V
Voltaje de entrada(recomendado)	7-12 V
Voltaje de entrada (limites)	6-20 V
Pines digitales E/S	14 (6 con salida PWM)
Pines de entrada analógica	6
Corriente continua para pines E/S	40 mA
Corriente continua para pines de 3.3V	50 mA
Memoria Flash	32 KB(ATmega328)
SRAM	2 KB(ATmega328)
EEPROM	1 KB(ATmega328)
Velocidad del reloj	16MHz

El esquema de pines es el siguente:

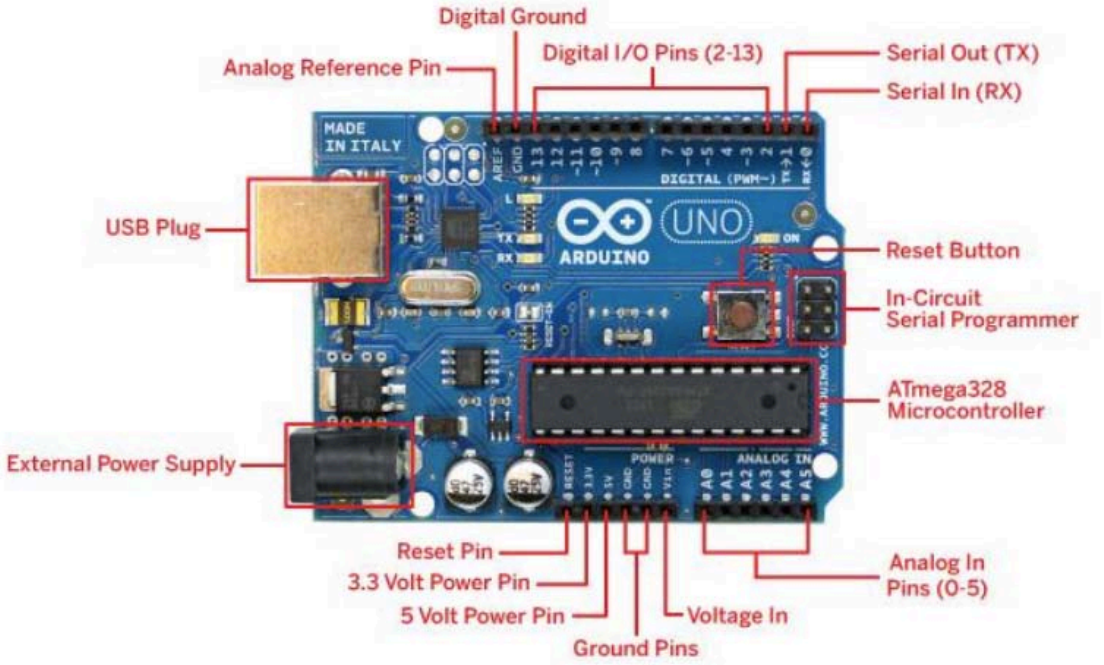


Figura 8 : esquema pines Arduino

Entorno de desarrollo

En cuanto al entorno de desarrollo de Arduino cabe a destacar que se trata de un entrono muy sencillo que facilita en gran medida la tarea del programador por lo tanto desarrollando un código sencillo y de fácil comprensión .

La plataforma Arduino tiene un lenguaje propio que está basado en C/C++ y por ello soporta las funciones del estándar C y algunas de C++. Sin embargo, es posible utilizar otros lenguajes de programación y aplicaciones populares en Arduino como Java, Processing, Python, Mathematica, Matlab, Perl, Visual Basic, etc. Esto es posible debido a que Arduino se comunica mediante la transmisión de datos en formato serie que es algo que la mayoría de los lenguajes anteriormente citados soportan. Para los que no soportan el formato serie de forma nativa, es posible utilizar software intermediario que traduzca los mensajes enviados por ambas partes para permitir una comunicación fluida.

Por otra parte se trata se un software multiplataforma que permite instalarlo en diferentes sistemas operativo como son Unix , Windows y Mac. Y se distribuye de forma gratuita para todos ellos a través de su propia pagina web.

Su interfaz grafica es muy amigable con el desarrollador como se puede ver en la siguiente figura y permite al programador tanto utilizar diferentes placas Arduino así como observar los estados de las salidas en cada momento gracias a su serial monitor.

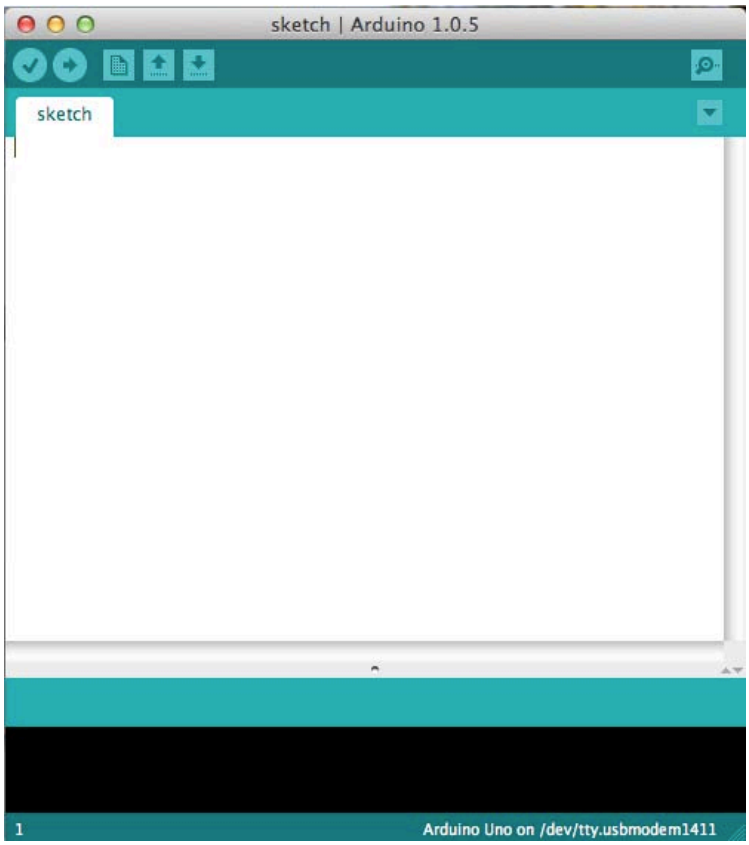


Figura 9: Entorno de desarrollo Arduino

Figura 10: símbolo Android

3. ANDROID: El Opened Software

Android es el sistema operativo de Google orientado a dispositivos móviles, fue lanzado al mercado en su versión 1.0 en octubre de 2008 y está basado en una versión modificada del kernel de Linux 2.6 el cual utiliza para controlar servicios del núcleo del sistema como pueden ser la seguridad, gestión de memoria o gestión de procesos. Actualmente, podemos encontrarlo presente en teléfonos móviles, PDAs, Tablets e incluso en Netbooks.

Es una plataforma de código abierto distribuida bajo la licencia Apache 2.0 por lo que su distribución es libre y posibilita el acceso y modificación de su código fuente. Inicialmente fue desarrollado por Google, para más tarde unirse a la Open Handset Alliance (de la cual, Google también forma parte) que está integrada por T-Mobile, Intel, Samsung, HTC o Nvidia entre otros. Sin embargo, Google ha sido la compañía que ha publicado la mayor parte del código fuente bajo la licencia Apache.

PLATAFORMA DE DESARROLLO

En Lo referido al ámbito del desarrollo de software para esta plataforma, se pueden desarrollar aplicaciones mediante el uso de el SDK o el NDK.

El NDK es un conjunto de herramientas que le permite implementar partes de su aplicación utilizando lenguajes de código nativo como C y C + +. Para ciertos tipos de aplicaciones, esto puede ser útil para que pueda volver a utilizar las bibliotecas de código existentes escritas en estos idiomas, pero la mayoría de las aplicaciones no es necesario el NDK Android.

En cuanto al SDK es un conjunto de herramientas que permite al desarrollador crear aplicaciones para la plataforma Android. Todo ello se puede integrar dentro del

entorno de desarrollo de Eclipse, donde se incluyen todas las APIs necesarias además de un potente emulador integrado para facilitar las pruebas de la aplicación y que nos ofrece una gran cantidad posibilidades.

En el caso del desarrollo de LEDSCRIPT hemos utilizado el entrono proporcionado por la SDK por su sencilla implementación así como su gran cantidad de documentación que proporciona.

EVOLUCION DE ANDROID

Desde su creación Android he tenido una continua evolución que le ha permitido convertirse en el sistema operativo para dispositivos móviles de referencia , pero a lo largo de su historia ha sufrido grandes cambios y continuas evoluciones que le han permitido ajustarse al mercado . por otra parte la gran acogida que ha tenido tanto por los fabricantes como por los usuarios le ha permitido ser el sistema operativo por referencia para dispositivos móviles junto a su competidor el iOS de Apple.

A continuación se detallan las evoluciones que ha sufrido el sistema operativo a lo largo de su historia.

Figura 11: Evolución de los sistemas operativos de Android

La arquitectura del sistema Android esta basado en un sistema diseñado por capas. Para ello utiliza el kernel de Linux 2.6 que le da acceso a la parte hardware de los dispositivos a la par que le permite ser compatible con muchos de los drivers creados para Linux. Esta arquitectura queda plasmada en el diagrama que encontramos en la siguiente figura.

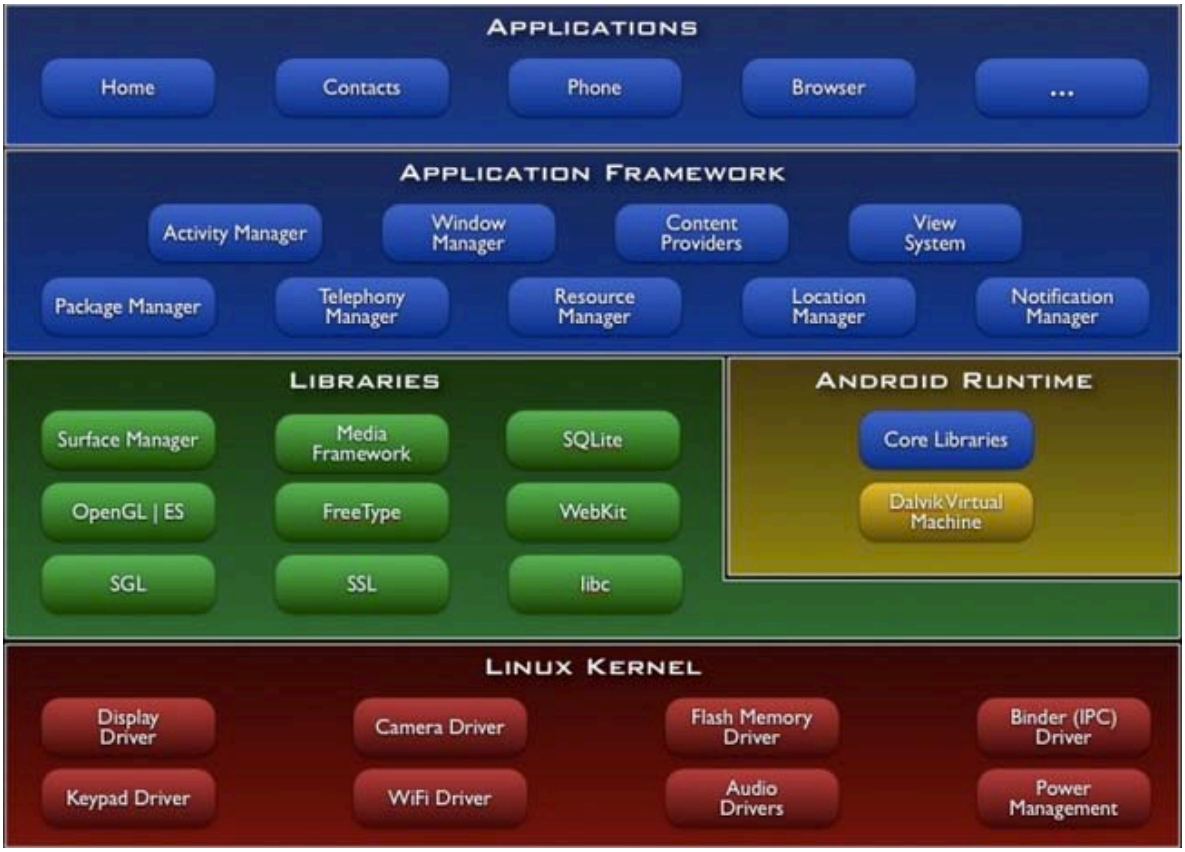


Figura 12: Arquitectura de Android

Kernel de Linux

La capa base es el núcleo de Linux. Todo el sistema operativo Android está basado en Kernel de Linux 2.6 con algunos otros cambios arquitectónicos realizados por Google. Es este Linux es el que interactúa con el hardware y contiene todos los controladores de hardware esenciales.

Librerias

Se trata de la parte de el sistema que incluye un conjunto de librerías en C y C++ que proporcionan la mayor parte de las funcionalidades presentes y que son utilizadas por varios de los componentes del sistema Android.

Android Runtime

En este caso estamos hablando del entorno de ejecución que esta compuesto por un conjunto de librerías que proporcionan la mayor parte de las funcionalidades java además de incluir la máquina virtual .

Framework

Estos son los bloques con los que nuestras aplicaciones interactúan directamente. Estos programas manejan las funciones básicas del teléfono, como la gestión de recursos, gestión de llamadas de voz, etc. Se considera que son las herramientas básicas con las que se construyen las aplicaciones.

Aplicaciones

Las aplicaciones son la capa superior de la arquitectura de Android . Por lo tanto aquí es donde se encuentran todas la aplicaciones pertenecientes a nuestro dispositivo móvil , tanto las aplicaciones preinstaladas en el sistema , como las que son creadas por los desarrolladores.

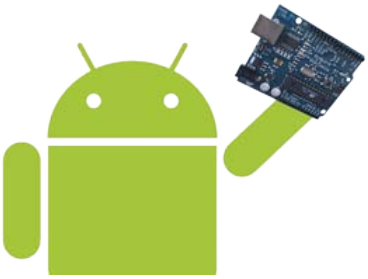


Figura 13: símbolo de Amarino

4. AMARINO : Donde se unen El Hardware y el Software Abiertos

Amarino es una aplicación Android y una librería Arduino, que permite comunicar una placa Arduino con el sistema operativo Android. Para enviar y recibir los datos de sensores o controlar los pines de la placa se utiliza un enlace bluetooth. La aplicación Android permite añadir eventos con su entorno visual.

Tiene como principal objetivo facilitar el desarrollo de interfaces tangibles a los smartphones y otros dispositivos Android, mediante la eliminación de varios de los pasos del desarrollo que son necesarios para su construcción. En particular, Amarino permite a los dispositivos móviles basados en Android comunicarse sin problemas con microcontroladores basados en Arduino.

El establecimiento de la comunicación se realiza mediante el uso de la tecnología Bluetooth. Para ello es necesario que el Smartphone disponga de un modulo bluetooth en su hardware propio. Por su parte Arduino tiene varias posibilidades , se puede adquirir un Arduino bluetooth, o bien se puede acoplar un modulo bluetooth a cualquiera de los dispositivos Arduino, teniendo en cuenta su compatibilidad.

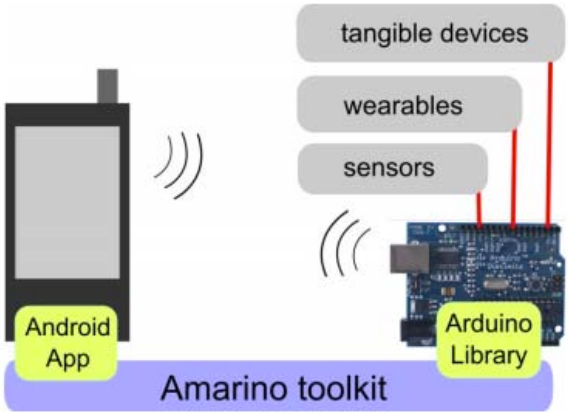


Figura 14: Esquema de Amarino

Por todo ello podemos diferenciar tres módulos dentro de la plataforma Amarino , que se describen a continuación.

Aplicación de Android

En lo que refiere a la aplicación Android cabe a destacar que se trata de una aplicación que hace las veces de interfaz entre el dispositivo Android y la placa Arduino. Dentro de la misma podemos destacar tres grandes módulos.

- El Administrador del dispositivo Bluetooth.
- El Administrador de Eventos.
- El monitorizador.

El primer paso que debemos seguir a la hora de conectar nuestros dispositivos es acceder al Administrador de dispositivos Bluetooth, el cual nos permite añadir nuevos dispositivos Bluetooth a nuestro terminal Android para así poder interactuar con los mismos. Para ello se deben seguir tres sencillos pasos, en primer lugar se debe acceder a la aplicación Amarino 2.0 y comenzar la búsqueda de nuevos dispositivo. A continuación se selecciona el dispositivo deseado, y finalmente se añade a nuestra lista , donde quedara vinculado a nuestro dispositivo. En la siguiente figura se muestra como se debe realizar.

Figura 15: Interfaz grafica de la Aplicación de Android

Otra de las particularidades que permite nuestra aplicación es que mantiene las conexiones abiertas aunque la interfaz sea cerrada. Esto es necesario para que nuestras aplicaciones que utilizan Amarino puedan funcionar con normalidad mientras nuestra aplicación de interfaz se ejecuta en el fondo. No obstante, la conexión permanece inactiva hasta que el usuario , mediante alguna aplicación, interactúa con ella, de forma que no se vean afectados los recursos de nuestro dispositivo tales como el procesador, la memoria, la batería, y otros tantos.

El segundo de nuestros módulos es el del Administrador de eventos que es el encargado de manejar las distintas aplicaciones que quieren acceder a nuestro dispositivo Arduino mediante nuestra plataforma Amarino. Todo ello permite que solo un evento sea trasmitido a nuestro dispositivo Arduino.

Nuestro manejador de eventos por su parte también tiene algunos eventos predefinidos por el Amarino plug-in bundle que permite sin necesidad de programar en nuestro dispositivo android probar la conectividad y realizar algunas pequeñas tareas básicas.

Algunos de los eventos predefinidos que podemos encontrar en nuestro manejados de eventos se pueden visualizar el la siguiente figura:

Plug-In Bundle Events	Description
Compass Sensor	Sends heading in degrees [0-359]
Accelerometer Sensor	Sends acceleration data in m/s2 [x,y,z]
Orientation Sensor	Sends orientation data in degrees [azimuth, pitch, roll]
Magnetic Field Sensor	Sends magnetic field data in micro-Tesla [x,y,z]
Phone State	Sends a message when the phone state changes [IDLE, RINGING, OFFHOOK]
Light Sensor	Sends a message when the ambient light level changes in SI lux units
Proximity Sensor	Sends the distance to the proximity sensor in centimeters
Battery Level	Sends the current battery level as soon as the phone's battery level changes
Time Tick	Sends a message every minute containing the actual minute
Test Event	Sends a test message every 3 seconds with random data [0-255]
Receive SMS	Forwards a received SMS to Arduino as String [only first 30 characters are sent]

Figura 16: Eventos predefinidos en la aplicación Android

Por último encontramos al monitorizador que es una herramienta de especial ayuda hacia el desarrollador pues le permite realizar una depuración del código introducido y así encontrar los posibles fallos que su propio código pueda contener.

El monitorizador también puede ser usado para enviar datos al propio Arduino de forma directa mediante el uso de flags como se puede apreciar en la siguiente figura:

Figura 17: Monitorizador de Android

Librería Meet Android

Se trata del homologo a la aplicación de Android pero en este caso para nuestro dispositivo Arduino. Por su parte esta librería lleva incluidos algunos pequeños programas que permiten probar algunas de las funcionalidades más sencillas que nos proporciona nuestra plataforma Amarino, como se puede apreciar en la siguiente figura.

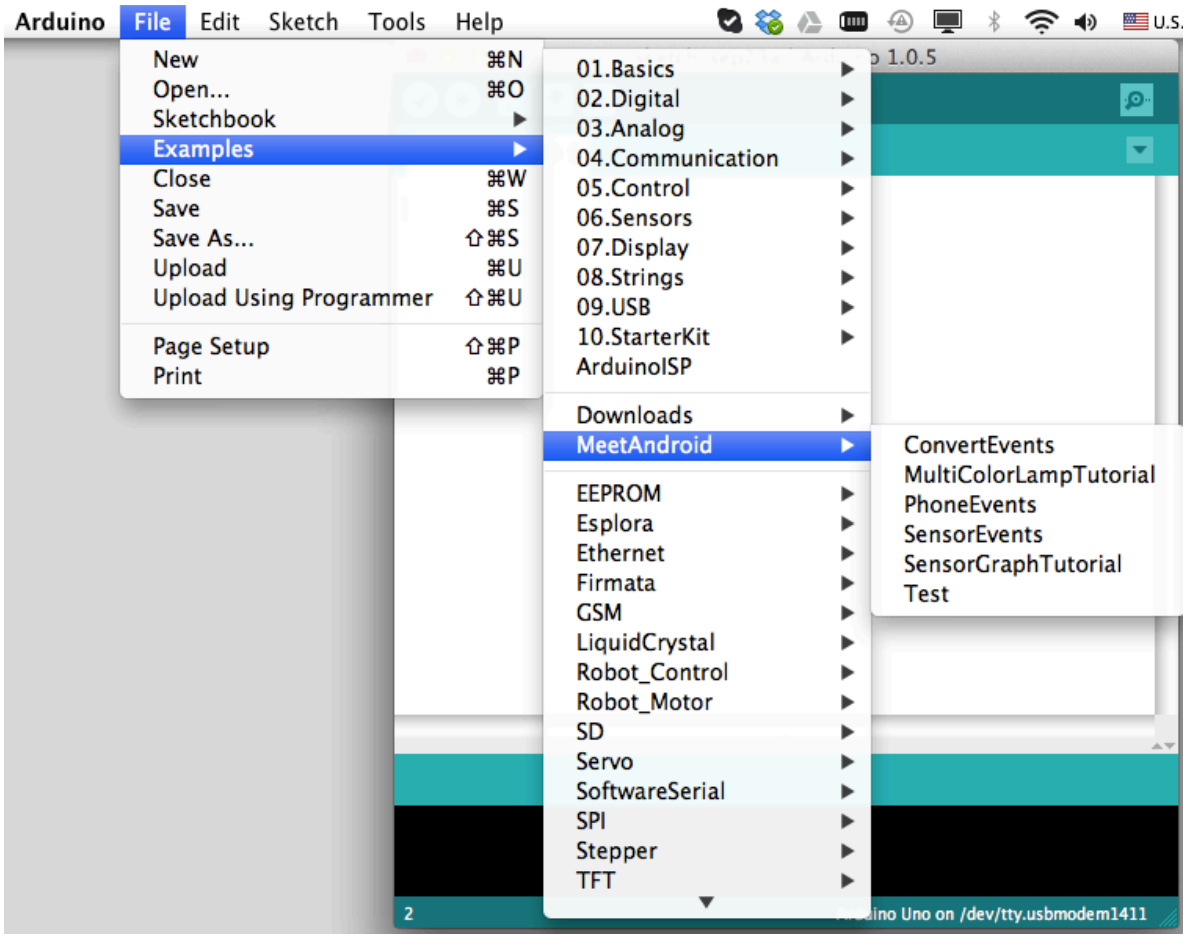


Figura 18: Libreria Amarino en el entorno de desarrollo Arduino

No obstante como toda librería Meet Android , nos ofrece una serie de métodos que permiten al desarrollador utilizar los métodos propios de la librería como son los siguientes.

Figura 19: Métodos de la librería meetAndroid

La API Amarino

Debido a que la aplicación de Android tiene algunas limitaciones se creo una API para suplir estas necesidades como pudieran ser la recepción de datos desde Arduino o interactuar con la pantalla del dispositivo móvil. Para suplir esta carencia esta Api implementa dos componentes que son los que constituyen la API de Amarino. En Primer lugar una clase llamada Amarino que contiene una serie de métodos estáticos, y en segundo lugar una interfaz llamada AmarinoIntent que contiene un grupo de constantes que se requieren para el uso de la API.

5. Diseño propio del proyecto

5.1 Diseño del Hardware

Dentro del proyecto LEDSCRIPT la parte del diseño del hardware ha sido la más importante , pues en la parte que se ha considerado de mayor interés, debido a que se disponía de menos material sobre el que tener una referencia de trabajo. Por todo ello se decidió realizar la parte del desarrollo del hardware en dos fases, una primera fase en la que se realizaría un prototipo del diseño final en la que se comprobarían que todos los componentes se acoplaban de forma correcta, y una segunda fase a posteriori en la que se diseña y construye el panel de LEDs con todos sus componentes.

Primera Fase : Prototipado

En esta primera fase se realizo una tarea de búsqueda de paneles similares al que nosotros queríamos desarrollar, pero en nuestro caso este tipo de paneles son poco frecuentes y fue una tarea ardua y difícil. No obstante toda búsqueda obtiene su recompensa , y encontramos un diseño que era similar a lo que se buscaba , que consistía en la unión de 4 paneles de LEDs prefabricados de 8x8 LEDs , que eran controlados mediante dos registros de desplazamiento de 8 bits y un Decodificador 4-16 en línea.

Después de la verificar que el diseño era compatible con las necesidades del proyecto se procedió a la adquisición de los chips necesarios para la construcción de nuestro prototipo. Los dos chips adquiridos fueron el decodificador CD74HC154E y el registro de desplazamiento de 8 bit SN74LS595N, que a continuación se describen:

El diseño encontrado se muestra en la siguiente figura:

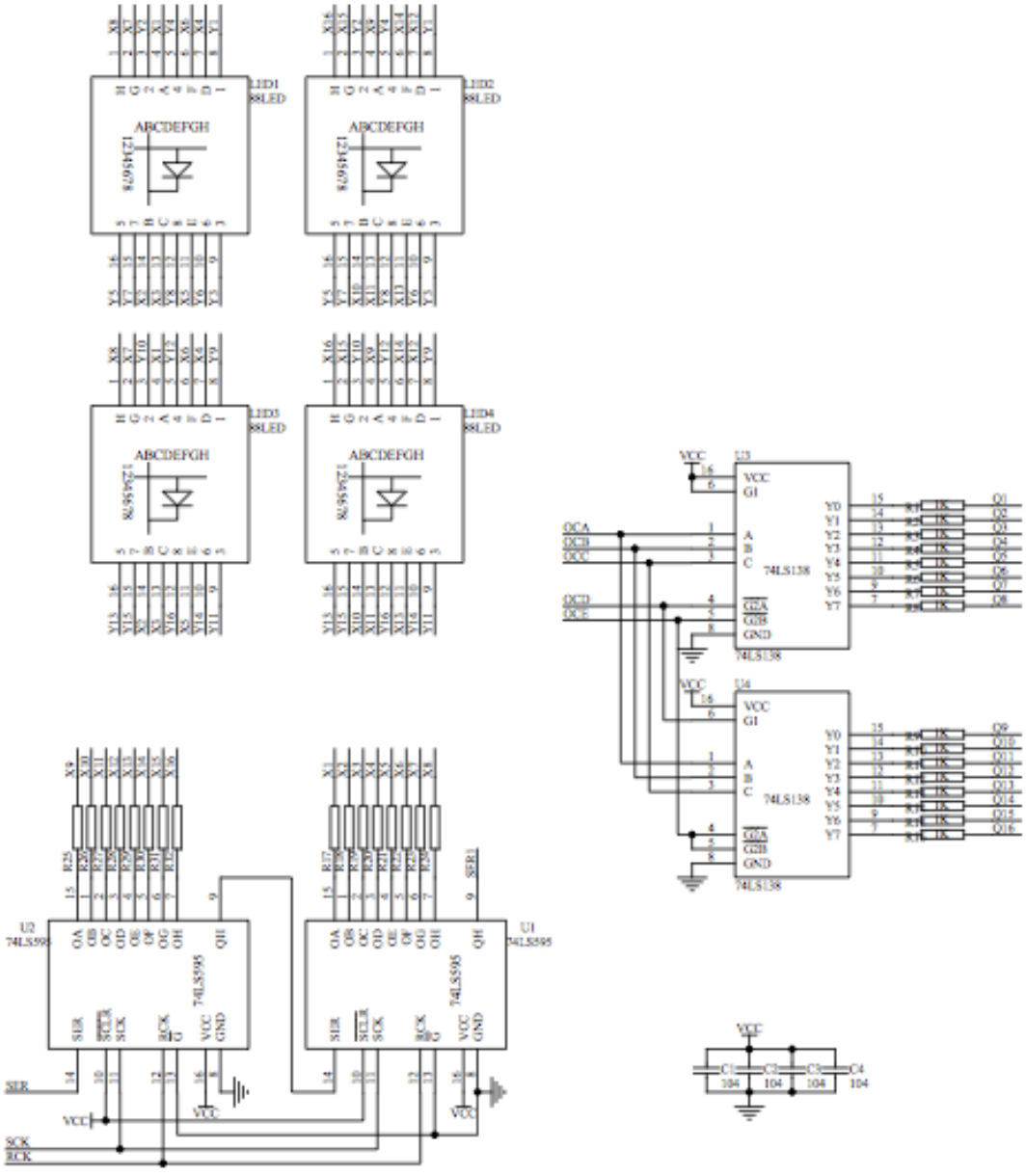


Figura 20: Diseño en el que se basa LEDSCRIPT

Decodificador CD74HC154E

Se trata de un decodificador/demultiplexor 4 -16, de alta velocidad basado en tecnología CMOS, su esquema de patillas lo podemos observar el la siguiente figura:

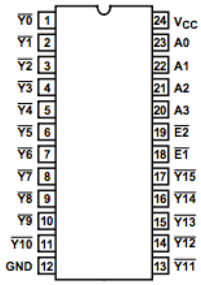


Figura 21: Esquema del decodificador

El patillaje se distribuye de la siguiente manera; en primer lugar consta de una entrada de corriente VCC a la que se le puede suministrar un voltaje máximo de 6V; a continuación se encuentran las entradas A0, A1, A2, A3, que son las entradas lógicas de nuestro decodificador; debajo de las anteriores encontramos las dos entradas de habilitación E0 y E1, que son activadas a nivel bajo y habilitan la decodificación de las entradas; Las salidas del decodificador son todas aquellas que se distribuyen desde Y0, Y1, ... Y14, Y15 de las que se habilitara a nivel bajo aquella salida que corresponda a la entrada cuando ésta esté habilitada; por ultimo nos encontramos con la masa GND. Para mayor comprensión de lo anteriormente explicado se incluye la tabla de verdad a continuación :

TRUTH TABLE

INPUTS						OUTPUTS														
$\overline{E1}$	$\overline{E2}$	A3	A2	A1	A0	$\overline{Y0}$	$\overline{Y1}$	$\overline{Y2}$	$\overline{Y3}$	$\overline{Y4}$	$\overline{Y5}$	$\overline{Y6}$	$\overline{Y7}$	$\overline{Y8}$	$\overline{Y9}$	$\overline{Y10}$	$\overline{Y11}$	$\overline{Y12}$	$\overline{Y13}$	$\overline{Y14}$	$\overline{Y15}$
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

Figura 22: Tabla de verdad del decodificador

Registro de desplazamiento de 8 bit SN74LS595N

Se trata De un registro de desplazamiento de 8 bit con entradas en serie y salidas en paralelo , es decir que se introducen las salidas deseadas una a una y posteriormente se despliegan todas las salidas a la vez, el patillaje se distribuye de la siguiente forma como podemos observar en la siguiente figura:

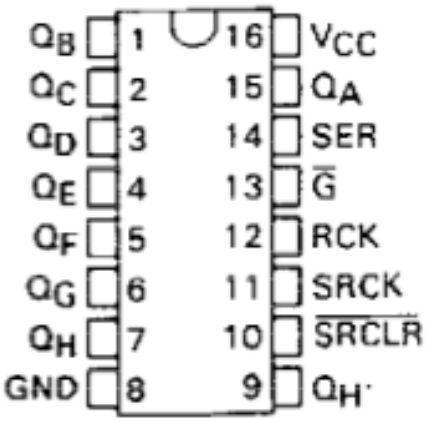


Figura 23: Esquema registro

En primer lugar disponemos de la toma de corriente VCC y la de masa GND, Las salidas son todas aquellas distribuidas entre la QA hasta la QH ambas inclusive , como entrada de serie se encuentra la patilla SER ; para el control de la recepción de datos mediante serie se encuentra la patilla SRCK que es la encargada de marcar los tiempos de las entradas; para el volcado de las salidas disponemos de la patilla RCK que vuelca los datos que contiene nuestro registro en ese momento; la patilla SRCLR es la encargada de resetear y limpiar los registros ; por ultimo la entrada G es la encargada de habilitar el funcionamiento de nuestro registro de desplazamiento .

A continuación podemos apreciar un fragmento del diagrama lógico de nuestro registro de desplazamiento:

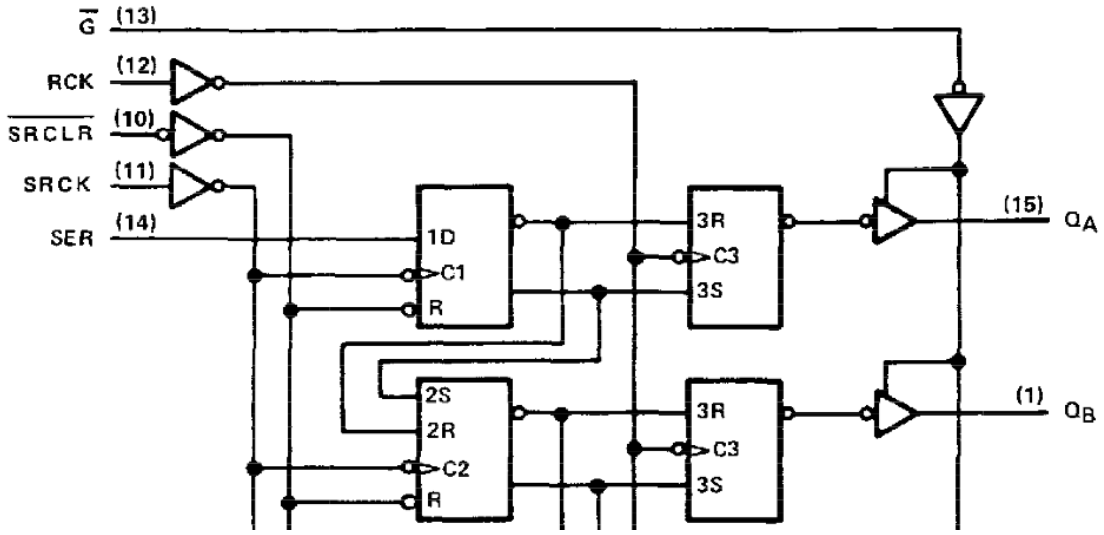


Figura 24: Diagrama registro

Una vez elegidos y adquiridos los componentes que eran necesarios para la construcción de nuestro dispositivo procedimos a la construcción y ensamblaje de un prototipo para comprobar que nuestros componentes realizaban la tarea para la que eran requeridos , para reducir el tiempo de trabajo decidimos construir una matriz de LEDs de 4 x 4 LEDs en la que simularíamos una matriz de 16 x 16 de forma que se conectaban las entradas y salidas 4, 8 , 12 y 16 simulando el funcionamiento de la matriz de 16 x 16. Una vez ensamblados todos los componentes se implemento un sencillo programa en el dispositivo Arduino que simulaba el funcionamiento de los LEDs , donde rápidamente se descubrió que la intensidad no era suficiente para la iluminación de la totalidad de los LEDs, por ello se tuvo que averiguar una nueva opción para resolver dicho problema.

Para solventar esta situación se opto por introducir un nuevo chip, en este caso el SN74LS244N un Buffer de 8 salidas , que incrementa la intensidad proporcionada por nuestro registro de desplazamiento y que solventa los problemas de luminosidad en nuestros LEDs.

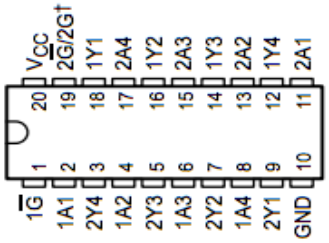


Figura 25: Esquema Buffer

Tras solventar nuestros problemas con nuestro nuevo chip , volvimos a comprobar que todo funcionaba correctamente, por otra parte a realización del prototipo mejoro las aptitudes técnicas en la soldadura de los chip de forma que en la fase de prototipado nos introdujimos en el mundo de la soldadura de microchips de una forma un tanto rudimentaria y aplicando los conocimientos y las técnicas aprendidas para la realización del panel final de forma mucho más limpia como se puede observar en la siguiente figura , una fotografía del panel prototipo:

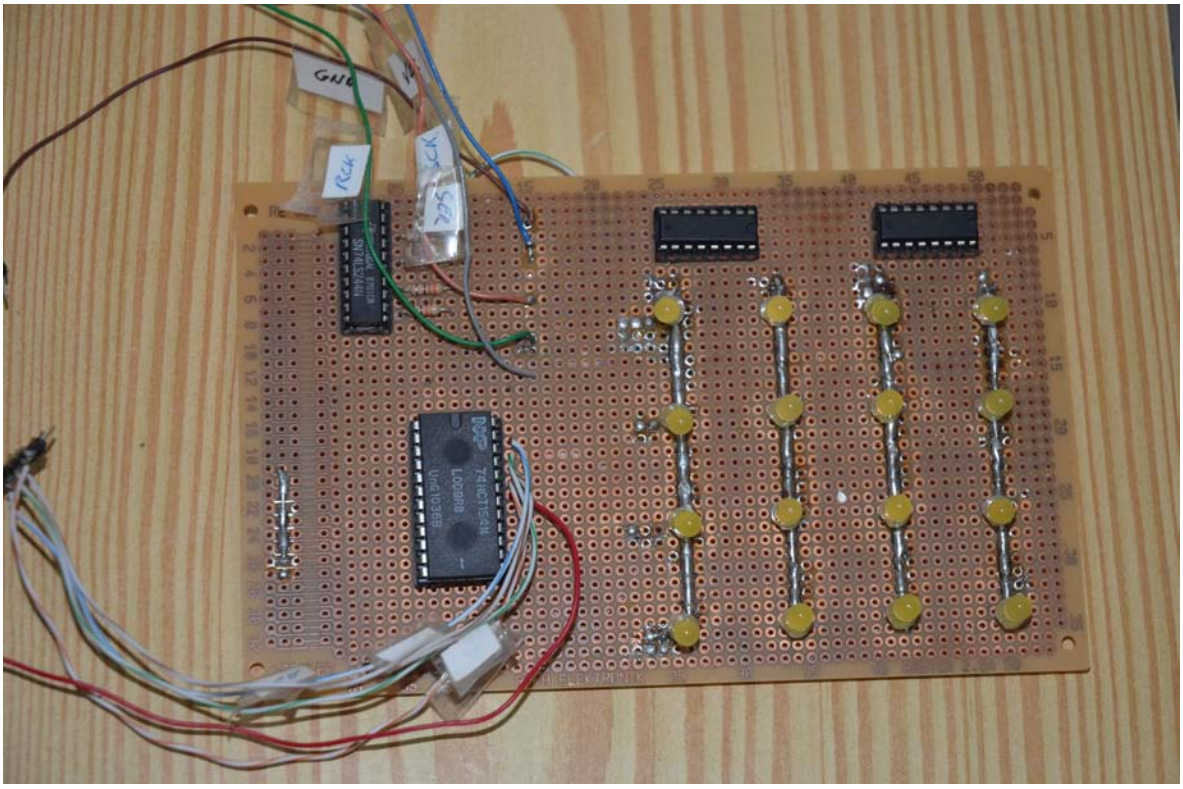


Figura 26: Fotografía del prototipo

Fase 2: Montaje del panel

En primer lugar cabe a destacar que esta fase fue bastante larga y costosa en su mayor parte debido a la gran cantidad de soldaduras que hubo que realizar. Para el diseño definitivo de nuestra placa de LEDs se decidió realizar un panel de 24 x 12 LEDs de forma que se insertaron en la placa un total de 288 LEDs , esta medidas no fueron elegidas a la ligera , pues previamente se realizo un estudio de las diferentes

posibilidades que habría para la inserción de textos legibles en nuestro panel, en primer lugar estábamos limitados por la placa perforada , pues nuestras dimensiones permitían un máximo 25 x 13 por lo que decidimos que el tamaño mínimo optimo para ser legible debía ser de 5 x 4 LEDs de esta forma restamos una fila a ambos lados para facilitar la labor de soldadura y de esta forma podríamos escribir con un espacio tanto vertical como horizontal 5 letras arriba y 5 letras abajo, por otra parte también se eligió este formato teniendo en cuenta el envío de datos a través de bluetooth desde nuestro dispositivo móvil de forma que también facilita el envío de los paquetes como posteriormente se explicara.

En este caso debido a que el numero de LEDs era diferente al de nuestro prototipo , fueron necesarios diferentes cantidades de chips para la realización de nuestra tarea, en este caso se necesitaron 3 registros de desplazamiento de 8 bits para cubrir las necesidades de los 24 LEDs horizontales así como otros 3 buffers de 8 salidas para proporcionar intensidad a los mismos. Por su parte el decodificador continuo siendo el mismo a pesar de que no se pudo optimizar el uso total de sus recursos , utilizando únicamente 12 de sus 16 salidas, pero como se ha indicado anteriormente estuvimos limitados por el tamaño de la placa perforada, a continuación se muestra como quedo nuestra pantalla de LEDs una vez todos los componentes fueron acoplados a la misma:

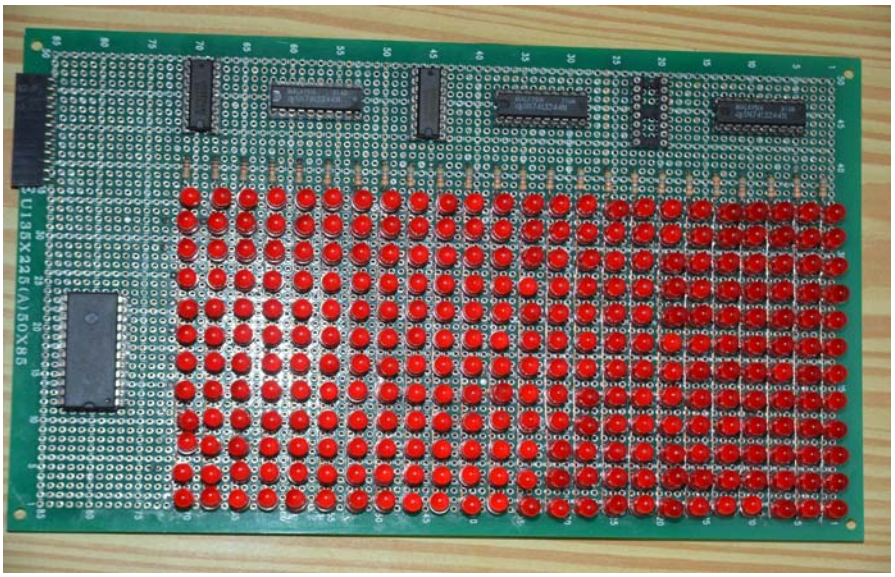


Figura 27: fotografía del panel terminado

5.2 Diseño del Software

En lo referido a la parte del software podemos dividir el trabajo en dos partes, el desarrollo del código para la reproducción de nuestros datos en la pantalla de LEDs mediante Arduino , y el desarrollo de la aplicación Android que recogería nuestra cadena de texto o patrón desde nuestro dispositivo Android.

Para el desarrollo del software que controla nuestro panel de LEDs se diseño un software lo mas sencillo posible que permitiera el fácil manejo y comprensión de la codificación así como un software que aplicara técnicas de refresco para que nuestro sistema fuera lo mas eficiente posible con un consumo de energía mínimo.

En primer lugar se decidió utilizar una técnica de refresco para de esta forma ahorrar la mayor energía posible , de forma que nuestras filas se refrescan en un periodo entre los 12 y los 15 ms , para ello se tuvo en cuenta que la velocidad del reloj de Arduino de 16 MHz , por tanto se dedujo que dependiendo del numero bits que se cargaran el proceso de carga total podría tardar entre 0 y 3 ms y que el tiempo necesario para el encendido de nuestros LEDs era de 1ms aproximadamente , y que el tiempo máximo que el ojo humano puede percibir el parpadeo era de unos 20ms por todo ello el refresco de nuestro panel de LEDs era factible.

Una vez comprobado que los requisitos técnicos no limitaban el funcionamiento se procedió a la programación del código que implementara las funciones que eran requeridas tales como la muestra de patrones o de palabras y letras, para ello se implemento un método principal como podemos apreciar a continuación:

```
byte BUFF[36];
    byte disrow= Ox00;
    int row;
    int Dpin[]={4,5,6,7};
    int OOE=2;
    int RCK=10;
    int SCK=9;
    int SER=8;
    int VCC=11;
    int GND=3;
void setup()
    {  pinMode(2,OUTPUT);
       pinMode(3,OUTPUT);
       pinMode(4,OUTPUT);
       pinMode(5,OUTPUT);
       pinMode(6,OUTPUT);
       pinMode(7,OUTPUT);
       pinMode(8,OUTPUT);
       pinMode(9,OUTPUT);
       pinMode(10,OUTPUT);
       pinMode(11,OUTPUT);
      digitalWrite(OOE,HIGH);
      digitalWrite(VCC,HIGH);
      digitalWrite(GND,LOW);
      digitalWrite(SCK,LOW);
    }
    void loop(){
    byte i;
      while(1){
            for(row=0;row<12;row++){           //recorro las 12 filas
                  digitalWrite(OOE,HIGH);
                  digitalWrite(RCK,LOW);
                  loadOneLine(row);            //cargo la fila row
                  sendPin(disrow);  //Envio la fila a cargar al Deco
                  digitalWrite(OOE,LOW);  //habilito la escritura Deco
                  digitalWrite(RCK,HIGH); // habilito la escritura  reg
                  delay(1);
                  disrow>>1;
                  }
            }
    }
```

La descripción del código anterior se corresponde al método principal y la configuración de nuestro sistema , la primera parte es la declaración e inicialización de variables y la segunda se corresponde al método principal del sistema en el que se recorren las 12 filas de nuestra matriz de LEDs de forma que mediante dos subrutinas *loadOneLine y sendPin* realizan la carga de cada una de dichas filas y mediante las operaciones de digitalWrite(OOE,LOW) y digitalWrite(RCK,HIGH); se habilitan las salidas de los registros de desplazamiento y el decodificador.

A continuación se detallan las dos subrutinas que implementan la carga de los registros de desplazamiento y el decodificador:

```
void loadOneLine( int row){
    int cont;
     byte s;
       for(cont=0;cont<3;cont++){
                       s= BUFF[row*3+cont];
                        for (int n=0;n<4;n++){
                             digitalWrite(SCK,LOW);
                  if((0x01&s) < 0x01)
                        {digitalWrite(SER,LOW);}
                        else
                        {digitalWrite(SER,HIGH);}
                  s>>=1;
                              digitalWrite(SCK,HIGH);
            }
       }
   }


  void sendPin(byte j){
            for (int n=0;n<4;n++){
                  if((0x01&j) < 0x01)
                        {digitalWrite(Dpin[n],LOW);}
                    else
                        {digitalWrite(Dpin[n],HIGH);}
            j>>=1;
       }
   }
```

En el caso de la primera subrutina , realiza la lectura de cada una de las filas representadas en el array BUFF , recogiendo 3 bytes correspondientes a los valores de 24 bits de cada uno de los LEDs de la fila . de forma que se van insertando uno a uno en los registros de desplazamiento.

Por su parte el método sendPin recoge la posición de la fila en la que nos encontramos y posiciona el decodificador para que seleccione nuestra fila.

La funcionalidad de nuestra matriz con estos métodos por el momento es autónoma , por lo que faltaría un método que recibiera nuestro array con los 36 bytes correspondientes a las 288 posiciones totales de nuestros LEDs.

En lo que corresponde al desarrollo de software en la aplicación para Android se ha realizado un estudio previo de las múltiples posibilidades que se podrían contemplar para la recepción de los datos. Finalmente se decidio que se ralizaria de dos formas como se puede apreciar en la figura siguiente que representa el diagrama que seguiría nuestra aplicación Android.

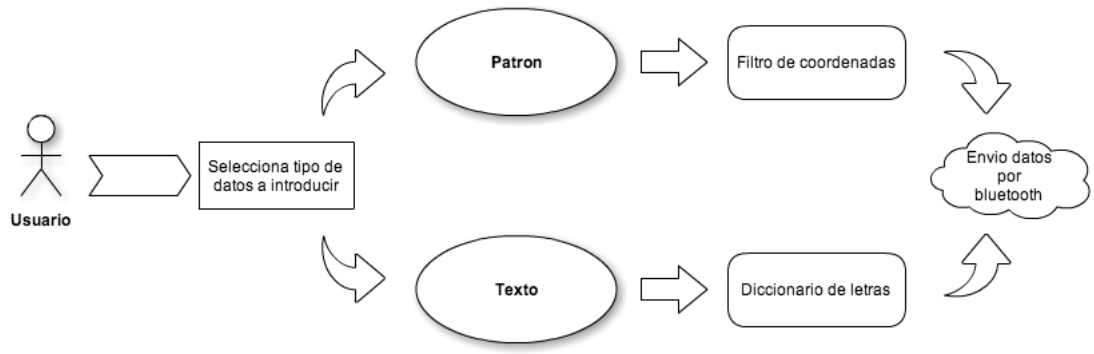


Figura 28: Diagrama aplicacion Android

Debido a diversos problemas tanto técnicos como físicos , el proyecto se quedo en esta fase no pudiéndose completar en su totalidad. No obstante el diseño de la aplicación Android se podría describir de la siguiente manera.

En primer lugar tendríamos una pantalla de inicio donde el usuario seleccionaría el tipo de datos que desea introducir , mediante texto o mediante un patrón de dibujo. Una vez seleccionado en el caso de el texto se introduciría la cadena de 10 caracteres , que podría ser de mayor tamaño con algunas modificaciones en el diseño de el código de Arduino, una vez obtenida la cadena se implementaría un diccionario que traduciría cada uno de los caracteres a un array de bytes que seria lo que se enviaría a nuestro Arduino. Por otra parte si decidimos introducir un patrón de dibujo , el proceso es diferente , en primer lugar se debe dibujar un patrón el la pantalla del dispositivo móvil , y una vez verificado , se realiza un codificado mediante una malla , que consiste en superpones una malla al patrón dibujado y en aquellos puntos de intersección en los que se haya dibujado un patrón se consideran activos y los que no lo tengan inactivos , de forma que mediante esta maya se crea un array de 36 bytes que es enviado a nuestro dispositivo arduino.

5.3 Conectividad vía Bluetooth

La conexión via bluetooth es la ultima parte del proyecto pero no por ello la menos importante , no obstante como se ha comentado anteriormente no se ha podido alcanzar esta fase , aun así también se realizo un estudio previo asi como unas pequeñas pruebas para verificar el funcionamiento de nuestros dispositivos.

En primer lugar cabe a describir el dispositivo bluetooth que se ha seleccionado, en nuestro caso elegimos el grove serial Bluetooth the seedstudio, que es un dispositivo bluetooth que fue diseñado para funcionar con Arduino.

Figura 29: Imagen del modulo Bluetooth

Nuestro dispositivo Bluetooth esta diseñado para ejercer las funciones de maestro como de esclavo por lo tanto puede tanto envía como recibir datos , tiene una velocidad de transferencia de hasta 3 Mbps a una frecuencia máxima de 2,4 GHz por lo tanto se trata de un dispositivo del tipo V2.0+EDR.

A pesar de no haber podido completar esta fase si se hizo un pequeño método que realizaba la recepción de los 36 bytes necesarios para el funcionamiento de nuestro panel de LEDs, el método es el siguiente:

```
void getBuffer(){


  for (int i=0; i<36; i++)
  {
      BUFF[i]=meetAndroid.getByte();
      meetAndroid.send(“ok”);
  }
}
```

Con este sencillo método se realizaría la recepción de los 36 bytes necesarios para construir nuestra matriz.

6. CONCLUSIONES

Este proyecto me ha ofrecido la oportunidad de mejorar mis aptitudes como ingeniero técnico informático especialmente orientado en el ámbito de la informática industrial. Con todo ello he podido aprender como se producen los hardware que posteriormente se programan , de forma que he podido seguir completamente todo el proceso de fabricación de un hardware propio desde la planificación y la elección de materiales hasta la implementación del código que controlaban los mismos , pasando por las fases de diseño, construcción y desarrollo.

He adquirido grandes conocimientos sobre las plataformas Arduino y Android, que son dos de los sistemas que permiten que los desarrolladores puedan trabajar de forma libre y gratuita sin necesidad de adquirir ningún tipo de licencia, así como en la arquitectura de la electrónica que me ha ayudado a comprender que el diseño de nuevos dispositivos hardware siempre es difícil y costoso , y nunca sabes lo que puedes encontrarte a la vuelta de la esquina.

Por otra parte me he demostrado que todas las metas son alcanzables y con ganas de trabajar y tesón se pueden conseguir todas las metas que te propongas.

Finalmente aunque no se ha podido concluir en su totalidad el proyecto LEDSCRIPT en el plazo fijado, en un futuro muy próximo será concluido de forma que se vea recompensado todo el esfuerzo que he invertido en este proyecto.

7. AGRADECIMIENTOS

En primer lugar quiero agradecer su gran apoyo a mi director de proyecto D. Ángel Rodas Jordá por su continua ayuda a lo largo de todo el proyecto aportando nuevas ideas y mejorando las propuestas, sin el cual no podría haberse hecho realidad este proyecto.

A mi familia, especialmente a mis padres y mi hermana por el gran esfuerzo que han realizado para poder otorgarme la oportunidad de estudiar la carrera que me gustaba a pesar de tener que separarme de ellos.

A mis amigos y a todos aquellos que me han apoyado a lo largo de te toda mi carrera y me han ayudado a superar mis metas.

8. BIBLIOGRAFIA

- http://arduino.cc/es/

  Página oficial de Arduino

- http://www.android.com/

  Página oficial de Android

- http://www.amarino-toolkit.net/

  Página oficial de Amarino

- http://www.emartee.com/Attachment.php?name=41903.pdf

  Esquema en el que se basa el prototipo de LEDSCRIPT

- Datasheets de los componentes en sus respectivas páginas web