# Universidad Nacional de La Matanza

Departamento de Ingeniería e Investigaciones Tecnológicas

# Sistemas Operativos Avanzados

# Informe final


Profesores:

Sebastián Barillaro

Esteban Carnuccio

Graciela de Luca

Gerardo Garcia

Waldo Valiente

Mariano Volker



| Apellido y Nombre | DNI |
|---|---|
| Bonaventura Gabriel | 38.695.713 |
| Casuscelli Camila | 38.891.527 |
| Cuassolo Facundo | 38.928.949 |

# Indice

1- Introducción ..... 3
1.1 Problemática ..... 3
1.2 Estrategia adoptada ..... 3
2- Decisiones tomadas y el criterio aplicado. ..... 4
3-Descripción detallada del funcionamiento ..... 5
4- Diagrama de componentes ..... 6
5. Diagrama de conexiones (circuito) ..... 6
6- Detalle de los componentes hardware y software utilizados: ..... 7
6.1 Hardware ..... 7
6.1.1 Arduino UNO ..... 7
6.1.2 Servo SG90 ..... 7
6.1.3 lcd 16x2 i2c ..... 8
6.1.4 Leds ..... 9
6.1.5 Sensor de proximidad hc-sr04 ..... 9
6.1.6 Botón ..... 10
6.1.7 Sensor de lluvia ..... 10
6.1.8 Sensor de luz fotosensible ..... 11
6.1.9 Bluetooth -HC06 ..... 11
6.1.10 Motorola Moto X2 ..... 11
6.2 Software ..... 12
6.2.1 Arduino 1.8.2 ..... 12
6.2.2 Android Studio 2.3.2 ..... 12
6.2.3 Windows 10 Education 64-Bits ..... 12
6.2.4 Android 6.0 Marshmallow ..... 12
7-Detalle de conexiones con sensores, actuadores y otros dispositivos, Protocolos de comunicación ..... 12
7.1 PWM Codificado ..... 12
7.2 Digital ..... 12
7.3 Analógica ..... 12
7.4 Serie I2C ..... 13
7.5 Serie ..... 13
8-Lenguaje y entorno de desarrollo ..... 13

# 1- Introducción

El documento tiene por finalidad informar sobre las decisiones y dificultades presentadas a lo largo del desarrollo del producto. Este, es un Automatizador de componentes de un vehículo. Donde el usuario podrá beneficiarse de este, de manera que no deberá interactuar con los comandos para activar o desactivar funcionalidades del vehículo que se podrían realizar por sí solas por medio de sensores y actuadores.
En el presente proyecto se realizan la solución de automatizar el limpiaparabrisas del vehículo, la automatización de los faros y, por último, brindarle una ayuda al momento de estacionar el vehículo, mostrándole la distancia en metros que tiene al objeto más próximo que tiene atrás, esto, al poner en marcha atrás el vehículo

## 1.1 Problemática

La problemática a la que nos enfrentamos es a la utilización de componentes extra a la hora de conducir un vehículo. Decidimos que acciones tales como activar un limpiaparabrisas o encender las luces del vehículo podían ser realizadas de forma automatizada.

## 1.2 Estrategia adoptada

Decidimos utilizar una estrategia donde en un primer momento (El desarrollo del Sistema Embebido), luego de seleccionar el producto a realizar. Fue la de separar las tareas por componentes físicos que deberíamos utilizar para la realización del producto final. De esta manera, cada uno de nosotros, venía a la siguiente clase con nuevo sketch para volcar en el Arduino e ir probando cada uno de estos componentes. Una vez examinada y entendida la funcionalidad de cada componente, procedimos a la integración de estos. Para que cumpla la funcionalidad final del producto. En este punto tuvimos conflictos ya que partes de código utilizadas para algunos componentes no eran compatibles con las de otros, como por ejemplo los delay. Esto nos llevó más tiempo de lo esperado, pero logramos resolver el problema y sacar adelante el Embebido funcionando.
Luego, al momento de desarrollar la aplicación para Android, decidimos las funcionalidades que íbamos a desarrollar en conjunto. Esta vez, al darnos cuenta de la incompatibilidad de código que tuvimos en la experiencia de Arduino, nos planteamos la idea de ir modificando un único proyecto. En este íbamos agregando modificaciones y cada uno de nuestro equipo tenía como objetivo una modulo diferente para la completitud de esta aplicación (como la conexión al BlueTooth, el manejo de hilos, las diferentes activity y la investigación de

sensores del dispositivo). Finalmente, conseguimos programar una app funcional, que interactúa de manera efectiva con el embebido anteriormente desarrollada.

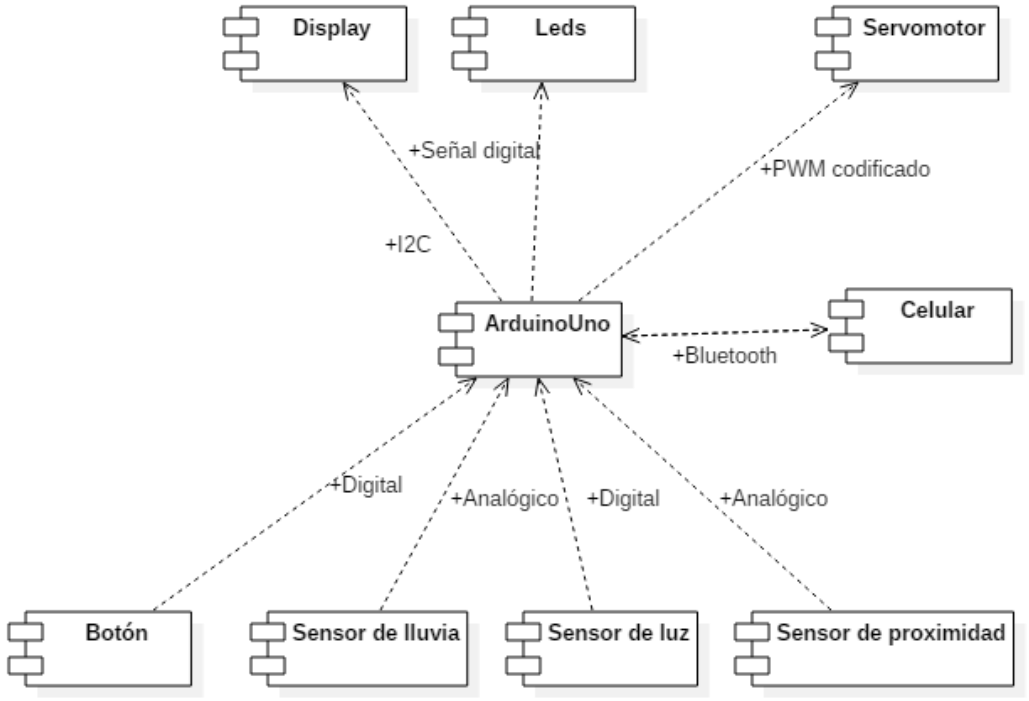
## 2- Decisiones tomadas y el criterio aplicado.

Como grupo decidimos investigar de forma individual y en conjunto sobre los distintos sensores y actuadores que podrían permitirnos resolver la problemática. En primer lugar, decidimos utilizar una placa Arduino D1 ya que nos permitía una conexión wi-fi de forma más sencilla por medio del módulo wifi esp8266 que viene integrado a esta. Al comenzar a hacer pruebas sobre los distintos sensores en el IDE Arduino nos encontramos con que la placa no funcionaba correctamente por lo que optamos por la Arduino Uno.
A la par discutimos sobre el tipo de comunicación que habría entre la placa y la app para Android. Al principio quisimos utilizar una conexión que consistía en Http response y http request, pero al encontrar dificultades con la placa Arduino D1 y optar por la placa Arduino Uno definimos que la comunicación se basaría en Bluetooth.
Por otro lado, a la hora de decidir qué sensores y actuadores utilizamos, la elección fue sencilla: Utilizamos sensores: Sensor de lluvia, sensor de luz, botón y sensor de proximidad (para detectar distancia entre el vehículo y un objeto) y para los actuadores elegimos: Servomotores para la función de limpia parabrisas, un display para indicar la distancia entre vehículo y objeto, y luces para alumbrar en caso de que no hubiese luz en el exterior.
Con respecto a Android, decidimos que sensores utilizar: Como primera instancia elegimos el sensor de acelerómetro para detectar un *shake,* sensor de luz y micrófono para detectar algunas frases que funcionaban como comando. Probando los sensores, descubrimos que el micrófono es muy sensible y que si hay ruido no detecta los comandos, por lo que decidimos dejarlo como un extra y elegir otro sensor: el sensor de proximidad. Además, tuvimos que decidir el diseño de la interfaz: por un lado, utilizamos una activity principal que muestra en forma de lista los dispositivos vinculados al celular y podemos elegir alguno de ellos para conectarse. Luego utilizamos otra activity que permite interactuar con el sistema embebido de forma que envía comandos y recibe datos del sistema. Para facilitar el funcionamiento y evitar procesamiento extra, definimos dos estados para la placa: modo automático o modo manual. El primer modo funciona de forma tal que los datos son tomados de los sensores propios de Arduino. El modo manual permite que el usuario a través de la app pueda enviar datos y accionar actuadores de la placa.
Por otro lado, tuvimos desacuerdos con la mensajería ya que no podíamos decidir si utilizar un servicio o un hilo. Elegimos un hilo porque tiene la propiedad de durar solo el tiempo que vive la app y no teníamos intención de permitir que la mensajería suceda aun cuando la app finaliza.

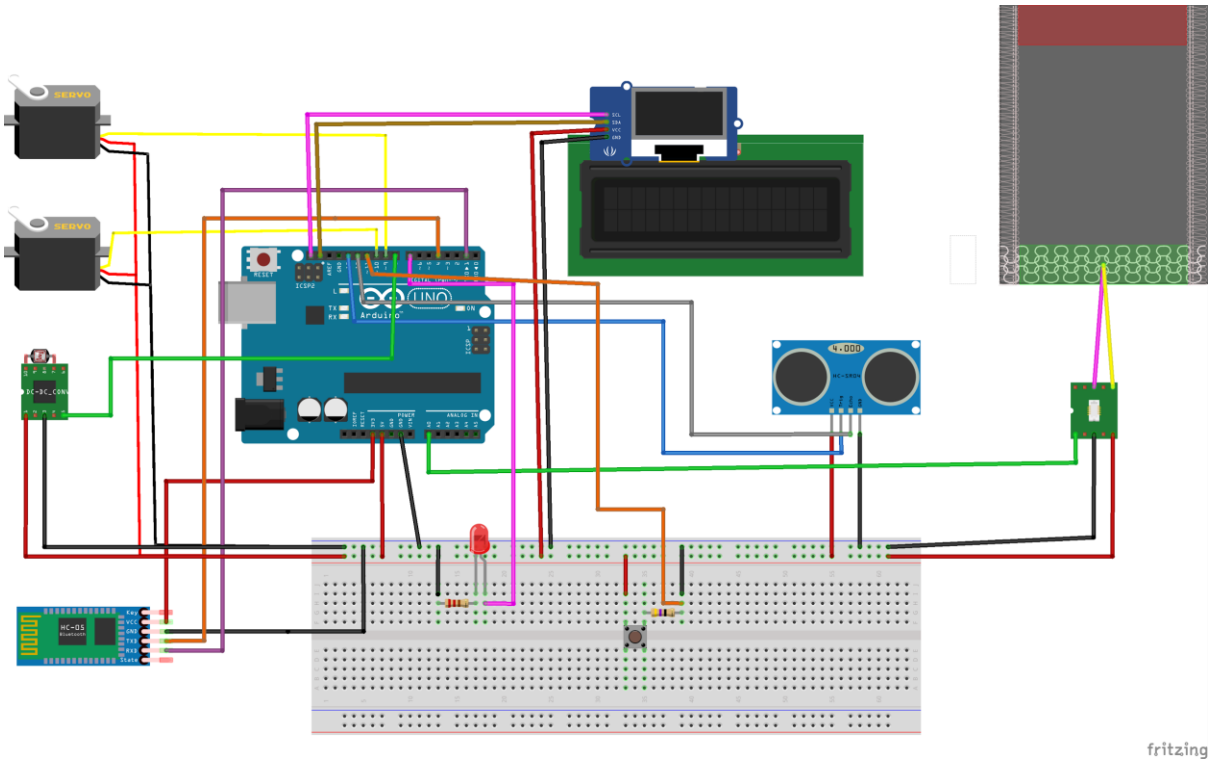
## 3-Descripción detallada del funcionamiento

La placa consiste en sensores: sensor de lluvia, sensor de proximidad, botón y sensor de luz. De forma tal que, si se detectara agua en el sensor de lluvia, se envían datos para que se active el servomotor delantero y si no se detecta agua y el servomotor está en funcionamiento, se apague. Si no se detecta luz en el exterior se encienden las luces del vehículo y si se detecta luz se apagan las luces. El botón es un sensor utilizado para simular la marcha atrás de un auto, si se presiona el botón cambiará la marcha. Es decir que, si el auto iba hacia delante, luego de presionar el botón se simula la marcha atrás y viceversa. Esto sirve para que comience a sensar el sensor de proximidad. La idea es que se asista a la hora de estacionar indicando la distancia entre el vehículo y el objeto que se encuentra detrás. También sirve para encender la luz "backlight" del display y comenzar a mostrar la distancia sensada; y mover el servo trasero en caso de que esté lloviendo. En conjunto con la app de Android el funcionamiento varía dependiendo del estado de la placa. Si se encuentra en modo automático funcionará de la misma manera que la detallada anteriormente; pero en caso de estar en modo manual el funcionamiento está dado por los sensores y botones del celular: un sensor de proximidad detecta el movimiento de la mano encima del celular para simular la marcha atrás y otro movimiento para desactivar la marcha atrás. El acelerómetro detecta el movimiento *shake* activando o desactivando el movimiento de los servomotores dependiendo su estado anterior. El sensor de luz de Android tiene la misma finalidad que el sensor de luz de Arduino: sensar la luz exterior. Por último, un detector de voz speech to text que permite de forma opcional activar la marcha atrás por medio de un comando. Además, la app es la encargada de establecer la conexión BlueTooth entre el teléfono y la placa.

## 4- Diagrama de componentes



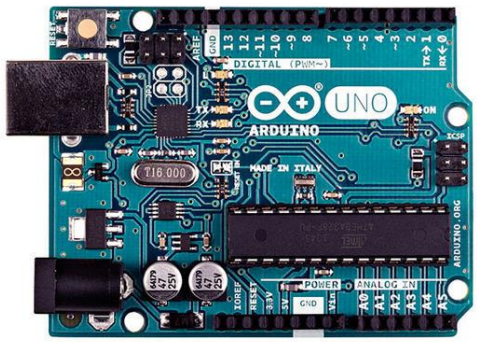

## 5- Diagrama de conexiones (circuito)



# 6- Detalle de los componentes hardware y software utilizados:

## 6.1 Hardware

### 6.1.1 Arduino UNO



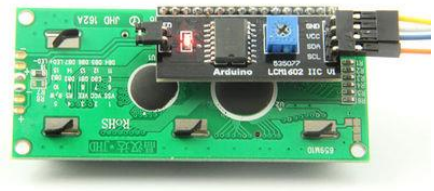

La placa de prototipado electrónica de código abierto basada en hardware y software flexibles y fáciles de usar.

### 6.1.2 Servo SG90

Para el manejo de estos se utilizó la librería <Servo.h>, con esta podemos manipular a los servos de manera más sencilla. Los servos como se sabe, se manejan por medio de PWM codificado. Lo que quiere decir que se determina la posición del servo, por medio del tiempo de un pulso. Esta librería con mandar la posición, en ángulos, realiza la conversión del tiempo que debe enviar el pulso para que se posicione en el ángulo deseado y además

proporciona el refresco para que el servo no vuelva al ángulo 0. Ya que, si no se mantiene el pulso de la posición repitiéndolo, el servo interpretara que debe estar en la posición 0.
Por medios de intervalos de tiempo utilizando la función millis (), logramos que cada 1 segundo, el servo se moviera de

### *6.1.3 lcd 16x2 i2c*




Con el LCD mostramos la distancia en metros al objeto más cercano de la parte trasera del vehículo. Este LCD dispone de un módulo I2C integrado, con este podemos comunicarnos con el display con este protocolo de manera serial. La librería utilizada fue <LiquidCrystal_I2C.h>. Esta permite la comunicación con el módulo integrado, que maneja el protocolo I2C. Este protocolo se maneja como maestro-esclavo. Donde solo el maestro puede iniciar la transferencia de datos. En nuestro caso, el Arduino toma el papel de maestro y el display el de esclavo.
El bus I2C, un estándar que facilita la comunicación entre microcontroladores, memorias y otros dispositivos con cierto nivel de "inteligencia", sólo requiere de dos líneas de señal y un común o masa.
La metodología de comunicación de datos del bus I2C es en serie y sincrónica. Una de las señales del bus marca el tiempo (pulsos de reloj) y la otra se utiliza para intercambiar datos.

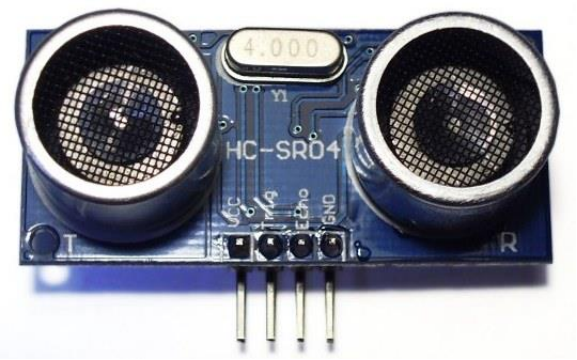
Descripción de las señales

SCL (System Clock) es la línea de los pulsos de reloj que sincronizan el sistema.
SDA (System Data) es la línea por la que se mueven los datos entre los dispositivos.
GND (Masa) común de la interconexión entre todos los dispositivos "enganchados" al bus.

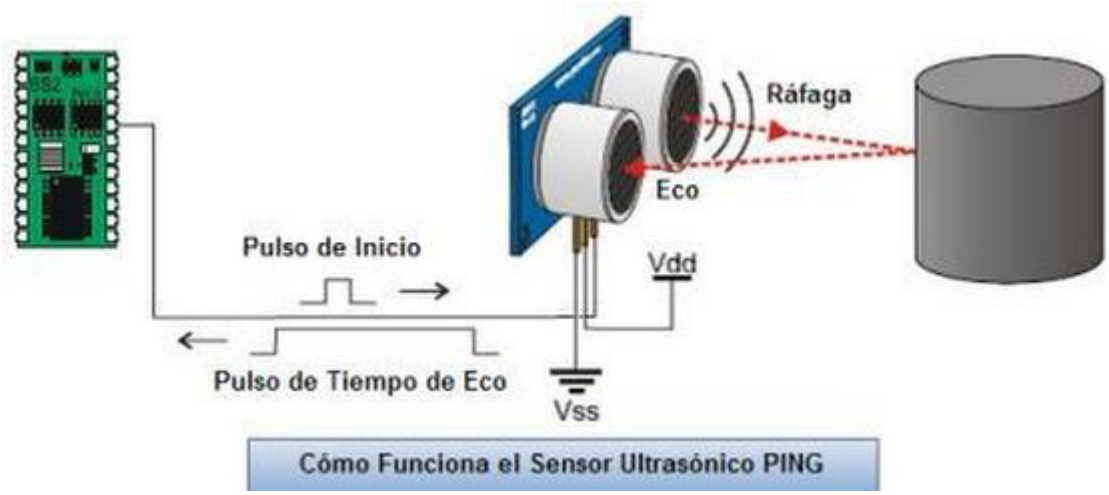
### 6.1.4 Leds

Los leds, que representan los faros de nuestro vehículo, son focos donde al brindarles la tensión determinada por el fabricante emiten luz. al regular la tensión el led puede brillar más o menos. En nuestro caso de estudio, el led esta en encendido (HIGH) o apagado (LOW)

### 6.1.5 Sensor de proximidad hc-sr04




El sensor de proximidad, se maneja con un trigger(disparador) de ultrasonido y un echo. De esta manera, el trigger envía una señal de ultrasonido, este revota contra el objeto más próximo y vuelve hacia el dispositivo que lo captura por el echo. Entonces este sensor mide en realidad, el tiempo que tarde en volver la señal del ultrasonido por medio del Clock y realizando una cuenta se puede calcular la distancia, ya que la señal de ultrasonido viaja a una velocidad constante.



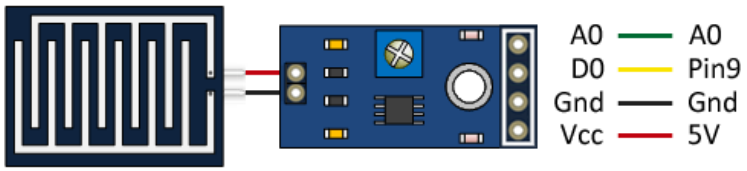

### 6.1.6 Botón

El botón es un mecanismo que deja pasar la tensión al ser presionado. Sin embargo, la implementación en Arduino no es tan simple como parece ya que se produce el efecto rebote.
Las láminas metálicas utilizadas en la construcción de las llaves/botones poseen, inherentemente, elasticidad. Por ello al intentar ponerlas en contacto se genera un choque que produce un movimiento en sentido contrario que aleja las láminas. Este proceso se repite hasta disipar la energía cinética adquirida por la lámina móvil.

### 6.1.7 Sensor de lluvia

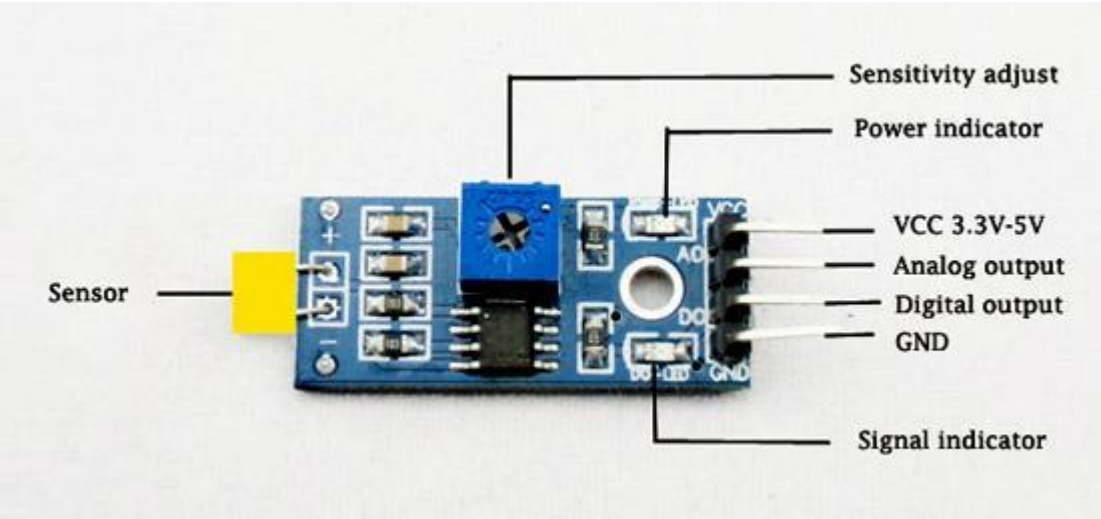
Este tipo de sensores detectan la presencia de lluvia por la variación de conductividad del sensor al entrar en contacto con el agua.
Se dispone de dos contactos, unidos a unas pistas conductoras entrelazadas entre sí a una pequeña distancia, sin existir contacto entre ambas. Al depositarse agua sobre la superficie, se pone en contacto eléctrico ambos conductores.
permite obtener la lectura tanto como un valor analógico como de forma digital cuando se supera un cierto umbral, que se regula a través de un potenciómetro ubicado en la propia placa.
Los valores analógicos medidos varían desde 0 para una placa totalmente empapada, a 1023 para una placa totalmente seca.

La salida digital dispara cuando el valor de humedad supera un cierto valor, que ajustamos mediante el potenciómetro. Por tanto, obtendremos una señal LOW en ausencia de lluvia, y HIGH con presencia de lluvia.
En el caso del sensor de lluvia la señal analógica carece de interés, ya que el sensor no dispone de la precisión necesaria para medir la cantidad de agua acumulada.




### 6.1.8 Sensor de luz fotosensible

Resistencia disminuye con el aumento de intensidad de luz incidente, logra pasar menos corriente por lo que se detecta como un 1, en caso contrario se detecta un 0.




### 6.1.9 Bluetooth -HC06

Es un módulo de Bluetooth cuya característica principal es que solo puede actuar como esclavo, por lo que solo se puede conectar a un dispositivo a la vez. Además, al ser esclavo dispone de un juego reducido de instrucciones.

### 6.1.10 Motorola Moto X2

Smartphone utilizado para la prueba de sensores de Android y la aplicación Android desarrollada.

## 6.2 Software

### 6.2.1 Arduino 1.8.2

Es la plataforma de desarrollo de Arduino, donde se escribe el código. Es el encargado de realizar el Cross compiler. Esto es, compilar en la máquina donde se desarrolla, para que se pueda ejecutar en el Arduino correspondiente.

### 6.2.2 Android Studio 2.3.2

Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android.

### 6.2.3 Windows 10 Education 64-Bits

Sistema operativo de Microsoft donde se instalaron todos los entornos para el desarrollo de ambas aplicaciones

### 6.2.4 Android 6.0 Marshmallow

Sistema operativo de Android, utilizado para realizar pruebas de la aplicación Android desarrollada

# 7- Detalle de conexiones con sensores, actuadores y otros dispositivos, Protocolos de comunicación

## 7.1 PWM Codificado

*Se utilizó este método de comunicación para el sensor ultrasonido y para el actuador servomotor*

## 7.2 Digital

*Se utilizó este método de comunicación para el sensor del botón y para los actuadores de los leds*

## 7.3 Analógica

*Se utilizó este método de comunicación para el sensor de lluvia*

### *7.4 Serie I2C*

*Se utilizó este método de comunicación para el actuador display, ya que este cuenta con un módulo I2C integrado para la comunicación con este.*

### *7.5 Serie*

*Se utilizó este método de comunicación para el modulo BlueTooth HC-06. Este método, nos permitió la comunicación de datos entre el Arduino y el dispositivo Android*

## *8- Lenguaje y entorno de desarrollo*

Para la parte de sistema embebido utilizamos el lenguaje Arduino con el IDE Arduino 1.8.2 e incorporamos una biblioteca externa llamada I2C Liquid crystal. Para la parte Mobile utilizamos el lenguaje Android con el IDE Android Studio 2.3.2. En este caso teníamos la opción de utilizar el IDE Eclipse, pero optamos por Android Studio por ser dedicado exclusivamente al desarrollo en lenguaje Android.

Fuentes
https://www.luisllamas.es/arduino-lluvia/