Universidad Nacional de La Matanza
Departamento de Ingeniería e Investigaciones Tecnológicas

Sistemas Operativos Avanzados

Internet of Things

Sistemas Embebidos y Android

“ParrillApp - Parrilla Inteligente”

Segundo Cuatrimestre - Año 2017

Días de Cursada: Lunes **Turno:** Noche Aula: 266

Docentes:

- Lic. Graciela de Luca
- Ing. Waldo Valiente
- Ing. Esteban Carnuccio
- Ing. Mariano Volker
- Ing. Sebastián Barillaro

Integrantes:

- Enciso Dure, Jorge Hernan DNI: 37.342.327
- González, Diego Andrés DNI: 38.662.215
- González Portillo, Maria DNI: 94.474.956
- Hermida Nicolás Emilio DNI: 36.923.103
- Di Giacomo Mauricio DNI: 38.050.774

Mail de contacto: enciso.hernan@yahoo.com.ar

Índice

Contenido

Índice 2
Objetivo del TP 3
Descripción general del sistema 3
Materiales Utilizados 4
Sistema Embebido 4
Aplicación Android 4
Alcance del Sistema 4
Sistema Embebido 4
Aplicación Android 5
Implementación 6
Sistema Embebido 6
Aplicación Android 7
Comunicación 8
Diagrama de Componentes 9
Diagrama de Conexiones 9
Problemas durante el desarrollo 10
Casos de prueba 11
Modo Automático 11
Modo Manual 12
Prototipo Terminado 13
Sistema Embebido 13
Aplicación Android 14

Objetivo del TP

Implementar los conocimientos obtenidos sobre IOT a lo largo de la cursada desarrollando un sistema que permita medir la temperatura y altura de una parrilla, pudiendo monitorearla y controlarla, utilizando Sistemas Embebidos y una aplicación Android.

Descripción general del sistema

Diseñamos una Parrilla “Inteligente” la finalidad de la misma es facilitar ciertos labores al momento de cocinar en la misma, notificando determinados eventos sin necesidad de estar cerca de la parrilla durante todo el tiempo que dure la cocción.

Entre las funcionalidades podemos mencionar que sensa la temperatura, y mide la distancia constantemente lo cual permite que el usuario conectándose desde la aplicación vea el estado de la parrilla (incluye temperatura, altura, entre otros valores medidos por los sensores de lluvia y proximidad).

ParrillApp posee dos modos: Automático y Manual

El **modo automático** trabaja sin la intervención del usuario, tomando altura y temperaturas máxima y mínima pre-seteadas, con estos parámetros como base levanta o baja la parrilla, y dispara las acciones correspondientes en caso de que se detecte lluvia (activa el LED) o proximidad (levanta la parrilla).

Para activar el **modo manual** se necesita una previa conexión vía Bluetooth, y luego se pueden configurar la temperatura (máxima y mínima) con la que se desea trabajar, una vez ingresados esos parámetros se puede volver al modo Automático.

O bien, una vez conectado vía Bluetooth el usuario puede subir o bajar la parrilla cuando desee, también puede apagar las alarmas que activa la misma tras detectar ciertos eventos.

Materiales Utilizados

Sistema Embebido

- Notebook Sony Vaio VPCEH30EL/B
- 1 x Arduino Uno R3 Original Atmel Atmega 328P
- 1 x Módulo Bluetooth Hc-06 Uart Ttl Arduino Pic Avr Esclavo
- 1 x Sensor Digital Temperatura Cable Sumergible Arduino (Ds18b20)
- 1 x Sensor infrarrojo para Arduino (1-CH)
- 1 x Sensor de Lluvia Raindrop Nivel de Agua Gotas Arduino Pic
- 1 x Sensor de UltraSonido (Us-100)
- 1 x LED 5mm RGB 4 patas 3.8V (L5RGB4P)
- 1 x Motor de RPM 50 (MR4-50 0804)
- 1 x Buzzer Miniatura 7-12V (HCM1212X)
- 1 x Integrado Driver de Motores Cuádruple Puente H L293 (L293-d)
- 1 x Gabinete Plástico 90X55X170 (H-21)
- 20 x Cable Protoboard M/M 20cm
- 1 x Protoboard

Aplicación Android

- Notebook Asus x45A
- Motorola G3
- Sensor de Proximidad
- Sensor de Luminosidad
- Acelerómetro

Alcance del Sistema

Sistema Embebido

1. El sistema embebido debe medir la temperatura de la parrilla.

2. El sistema embebido debe implementar rangos de temperatura (Mínima y Máxima).

3. El sistema embebido debe medir la distancia de la parrilla a las brasas.

4. El sistema embebido debe utilizar diferentes colores del LED RGB, para los diferentes eventos que analiza la parrilla:
   - Azul: Detecta lluvia
   - Rojo: Detecta temperatura superior a la temperatura máxima o inferior a la mínima

5. El sistema embebido debe accionar el Buzzer:
   - Cuando detecte la proximidad el sensor infrarrojo

6. El sistema embebido debe accionar el motor en las siguientes condiciones:
   - Subir la parrilla (hasta una altura máxima) cuando la temperatura supere la temperatura máxima.
   - Bajar la parrilla (hasta una altura mínima) cuando la temperatura esté por debajo de la temperatura mínima.
   - Subir la parrilla (hasta una altura máxima) cuando detecte que alguien se acerca a través del sensor infrarrojo.

Aplicación Android

Se define el modo manual, que trabaja en cooperación con el Arduino:

1. La aplicación Android debe recibir e informar la temperatura a través de una comunicación vía Bluetooth.

2. La aplicación Android debe recibir e informar el estado de los actuadores (Buzzer y LED) a través de una comunicación vía Bluetooth.

3. La aplicación Android debe recibir e informar si se sensa lluvia o si se detecta proximidad con el sensor infrarrojo a través de una comunicación vía Bluetooth.

4. La aplicación Android debe poder indicar los valores de temperatura mínimo y máximo a utilizarse en el sistema embebido.

5. La aplicación Android debe poder indicar la altura a mover de la parrilla en cm.

6. Cuando el celular está en posición “horizontal” la aplicación Android debe poder apagar el Buzzer utilizando el sensor de proximidad.

7. Cuando el celular está en posición “vertical” la aplicación Android debe poder apagar el LED utilizando el sensor de luz.

8. Cuando el celular está en posición “vertical” la aplicación Android debe poder indicar al Arduino en qué sentido mover la parrilla (subir o bajar) con el acelerómetro.

9. La aplicación Android debe informar al usuario cuando la temperatura sea “Baja”, “Alta” o “Correcta”.

Implementación

Sistema Embebido

Configuración del módulo Bluetooth

Inicialmente configuramos el Módulo Bluetooth mediante los comandos AT (se configuraron el Nombre, la Velocidad de transferencia y el PIN). Luego, mediante la biblioteca “SoftwareSerial.h” mapeamos los pines Tx y Rx a pines digitales para poder emitir y recibir información de los sensores a la Aplicación.

Implementar Sensor de Temperatura

Para este sensor, utilizamos dos librerías (“OneWire.h”, que sirve para enviar y recibir datos por el único cable de datos del sensor de temperatura, y “DallasTemperature.h”, la cual permite la implementación de funciones como la de obtener la temperatura en °C).

Implementar Sensor de Proximidad Infrarrojo

Para éste sensor, únicamente obtenemos los datos de los pines conectados mediante la función “analogRead”. Ésta función nos determina que si se lee un valor mayor al seteado por el potenciómetro del sensor, hay proximidad a un objeto.

Implementar Sensor de Ultrasonido

En cuanto a éste sensor, obtenemos mediante el pin “Echo” el tiempo que tarda el sonido en llegar desde el pin “Trigger”. Una vez hecho esto, implementamos cálculos matemáticos para determinar la distancia en la que se encuentra el sensor con el carbón de la parrilla.

Implementar Sensor de Lluvia

Hay dos tipos de implementaciones posibles en cuanto a éste sensor: puede ser utilizado como sensor analógico (el cual mide en un rango de 0 a 1023 la presencia de lluvia) o como un sensor digital (el cual nos informa si estamos en presencia de lluvia o no).

Básicamente, lo implementamos con una configuración digital, ya que nos interesa saber si hay o no lluvia (implementando la función “digitalRead(pinLluvia)” sabemos si hay o no lluvia, y actuamos en base a ésta información).

Implementar Actuador Buzzer

A través de éste actuador informamos al usuario que se detectó la presencia de un objeto en las proximidades de la parrilla.

Implementamos la función “analogWrite(pin, intensidad)” por la cual al pin conectado al buzzer se le ingresa un valor de intensidad por el que el sonido es obtenido.

Implementar Actuador LED RGB

Simplemente, mostramos un color titilante ROJO en el caso de que la temperatura de la parrilla esté fuera de sus rangos mínimo o máximo, y un color titilante AZUL en el caso de que estemos en presencia de Lluvia.

Para implementación de éste actuador, utilizamos la función “analogWrite(pin, intensidad)”, por la cual cada pin conectado a los colores del RGB emite un color (por su intensidad).

Implementar Motor

Utilizamos el Puente H L293D mediante el cual hacemos girar el motor para subir, bajar la parrilla o para que frene mediante funciones que le entregan voltaje a las entradas del motor, entregando voltaje a una entrada gira en un sentido y entregando a la otra entrada, gira en el sentido contrario y al no entregarle voltaje a ninguna de las dos, se frena quedando en estado de reposo.

Aplicación Android

Creación de Activities

Se crearon 3 activities (interfaces de usuario) dentro de la aplicación. Una funciona como pantalla principal, en la cual se muestran los dispositivos bluetooth que están emparejados con el teléfono (en caso de que el bluetooth este desactivado, se pide su activación), permitiendo su selección. Una vez seleccionada la parrilla del listado, se procede a conectar con dicho dispositivo y a abrir la segunda activity, en la cual se realiza todo el control manual, ya sea utilizando los sensores, como indicando por escrito el rango de temperatura con el cual se trabaja. La tercera activity es una activity cuya única finalidad es mostrar una pantalla de ayuda, en la cual se explica el funcionamiento completo de la aplicación.

Implementación Sensores

Implementando la interfaz **SensorEventListener**, se realizó un override del método onSensorChanged(SensorEvent event). Dentro de este método, se identificó que sensor fue el que recibió un cambio en sus valores medidos, y acorde a esto, se realiza un llamado a distintas funciones para analizar el valor y realizar el funcionamiento esperado.

Implementar Acelerómetro

Lo primero que realizamos con el acelerómetro, fue distinguir la posición del teléfono de vertical a horizontal, en cuyo caso se mide o ignoran los otros sensores (El sensor de Luz es ignorado cuando el teléfono está en horizontal, lo opuesto ocurre con el de Proximidad). Una vez identificada la posición, se procedió a llamar a los métodos correspondientes según sea el caso.

Además, si se sensa que el teléfono realizó un movimiento de shake (es decir, si inclinó el teléfono en algún sentido), se verifica que sea en los sentidos que corresponden al envío del mensaje para que el Arduino mueva o no el motor. Separando el movimiento en dos, uno hacia la izquierda (bajar motor) y otro hacia la derecha (subir motor).

Implementar Sensor de Luz

Una vez detectado que el teléfono se encuentra en posición vertical, cada vez que varía el valor sensado del sensor de proximidad, realizamos un análisis del valor para corroborar que haya sobrepasado un rango especificado. En caso afirmativo, tomamos ese cambio de luz como mensaje, para que se envíe al Arduino la orden de apagar el LED.

Como las condiciones de luz dependen del ambiente en el que uno se encuentra, se agregaron tres niveles de luz a elegir, modificando el rango sobre el cual se analizan los valores de luz obtenidos.

Implementar Sensor de Proximidad

Una vez detectado que el teléfono se encuentra en posición horizontal, cada vez que se detecta una variación recibida en los valores del sensor de Proximidad, se realiza un análisis para corroborar que, en efecto, la variación haya sido la necesaria para indicar que se debe enviar la orden de apagar el Buzzer al Arduino. Esto se realiza comparando el valor obtenido con un rango específico (que como no varía de acuerdo al ambiente, es un rango estático y predefinido).

Comunicación

Para realizar la comunicación con el sistema Arduino, se implementó una conexión mediante un canal Bluetooth. Cada vez que se quiera utilizar la parrilla en modo manual (es decir, manejarla con el teléfono y la aplicación Android), se elige el dispositivo bluetooth y se comienza la conexión. Se abren los sockets pertinentes, y en caso de que se genere la conexión con éxito, se inicia un hilo encargado de leer constantemente el canal bluetooth a la espera de mensajes por parte del Arduino. En caso de recibirlos, un handler se encarga de leer los datos, analizarlos y descartar los mensajes fallidos. Aquellos que sean pertinentes a la aplicación, serán informados en la activity del Control Manual.

A su vez, cada vez que se requiera enviar información desde el Android al Arduino, se utilizará un método de escritura que pertenece al hilo en ejecución, el cual escribe en el stream entre ambos dispositivos.

Se utilizaron las clases **Thread**, **BluetoothAdapter**, **BluetoothSocket**, y **Handler** como medios principales para la realización de la comunicación desde Android.

Diagrama de Componentes

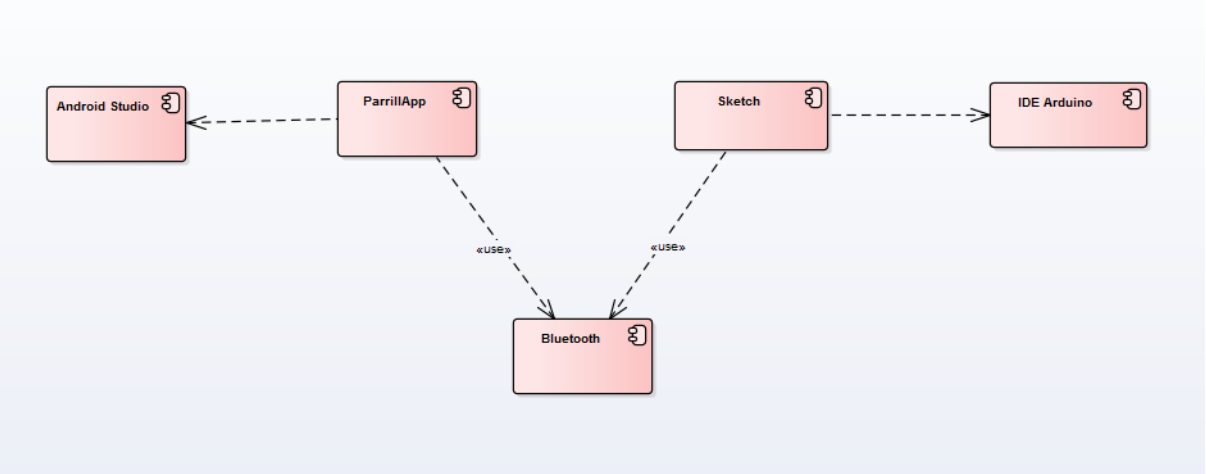
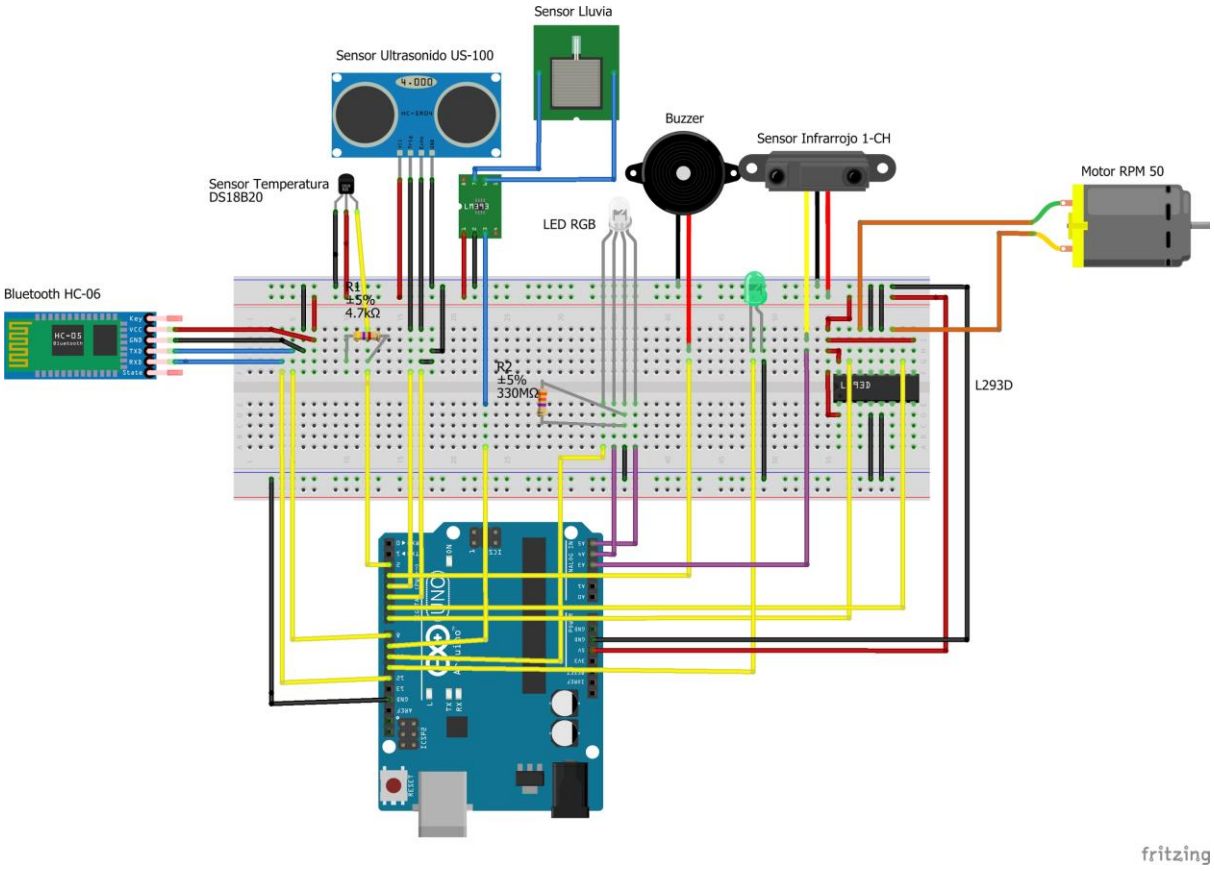


Diagrama de Conexiones



Problemas durante el desarrollo

1. **Varios sensores utilizando el mismo actuador en Arduino**
   Descripción: Ejemplo: Los sensores de Temperatura, lluvia y proximidad encendían el buzzer. Si detectaba temperatura lo prendía, pero como no detectaba lluvia lo apagaba.
   Solución: Tuvimos que utilizar flag para cada sensor y agregar una lógica adicional para que si un sensor está accionando un actuador, otro sensor no cambie su estado

2. **Sensor de ultrasonido media "0" cuando la distancia no era 0.**
   Descripción: El sensor de ultrasonido hacia contacto con el material metálico de la parrilla, esto generaba valores indeseados, haciendo que la medición de la altura de la parrilla no sea la correcta, indicando en muchas ocasiones que su altura era de "0" cm, cuando en realidad tenía un valor distinto.
   Solución: Aislamos con un cinta los orificios en los que se coloca el sensor para que no haga contacto con el metal.

3. **Arduino no reconocía mensajes enviados por Android**
   Descripción: Se enviaban mensajes desde la aplicación y el Arduino no recibía correctamente la cadena de texto enviada.
   Solución: Hubo que incluir un *"\n"* al final de cada línea enviada desde Android, para que desde Arduino se pueda detectar el final de la línea.

4. **De vez en cuando, en la conexión por Bluetooth se enviaban mensajes con caracteres que no correspondía o faltaban caracteres.**
   Descripción: Se enviaba por ejemplo "T25-40" y se recibía "T25?40" donde se indicaba la temperatura mínima y máxima a establecerse.
   Solución: Se implementó un método de filtrado de mensajes en Arduino, para ignorar los mensajes no correspondientes.

5. **El motor no frenaba, una vez que se accionaba seguía moviendo enroscando y desenroscando las cadenas**
   Descripción: Al activarse el sensor Infrarrojo o temperatura, la parrilla tenía que subir hasta la altura máxima y el motor tenía que frenar al llegar a dicha altura pero este no frenaba y se pasaba de
   Solución: Se modificó la función moverMotor() a la cual se agregaron cotas (superior e inferior).

6. **Los valores para los cuales se utiliza el Sensor de Luz son cambiantes de acuerdo a la ubicación del teléfono**
   Se implementaron tres rangos seleccionables por el usuario, como niveles de luz bajo, medio y alto, filtrando los cambios de luz no correspondientes.

Casos de prueba

Modo Automático

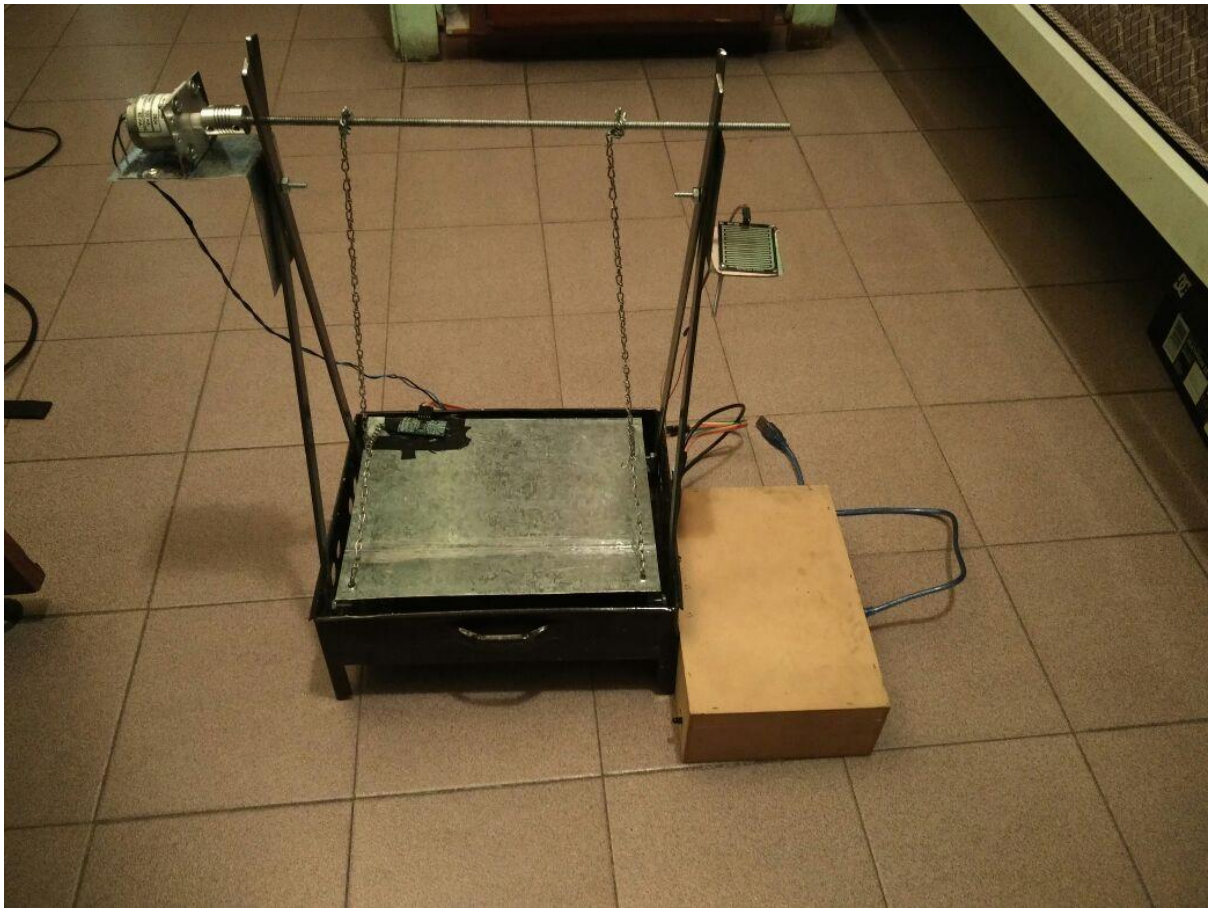
Casos	Resultado esperado
Caso 1: Detecta temperatura mayor a la temp máxima (cada 1 seg)	
Temperatura actual: 40	Sube la parrilla hasta que la temperatura actual este dentro del rango entre la max y la min, sino sube hasta la altura máxima
Temperatura max: 35	Enciende LED rojo
Temperatura min: 20	
Caso 2: Detecta temperatura menor a la temperatura mínima (cada 1 seg)	
Temperatura actual: 15	Baja la parrilla hasta que la temperatura actual este dentro del rango entre la max y la min, sino baja hasta la altura mínima
Temperatura max: 35	Enciende LED rojo
Temperatura min: 20	
Caso 3: Detecta lluvia (cada 1 seg)	
	El LED azul se enciende
Caso 4: Proximidad detectada	Se enciende el Buzzer
	Sube la parrilla hasta la altura máxima
Caso 5: Mientras está subiendo la parrilla por temperatura detecta lluvia	
	Enciende LED azul
	Notificación al teléfono desde la aplicación
Caso 6: Mientras está subiendo por temperatura detecta proximidad	
	Sube la parrilla hasta la altura máxima
	Enciende buzzer
Caso 7: Detecta lluvia + proximidad	
	Sube la parrilla hasta la altura máxima
	Enciende LED azul
	Notificación al teléfono desde la aplicación

Modo Manual

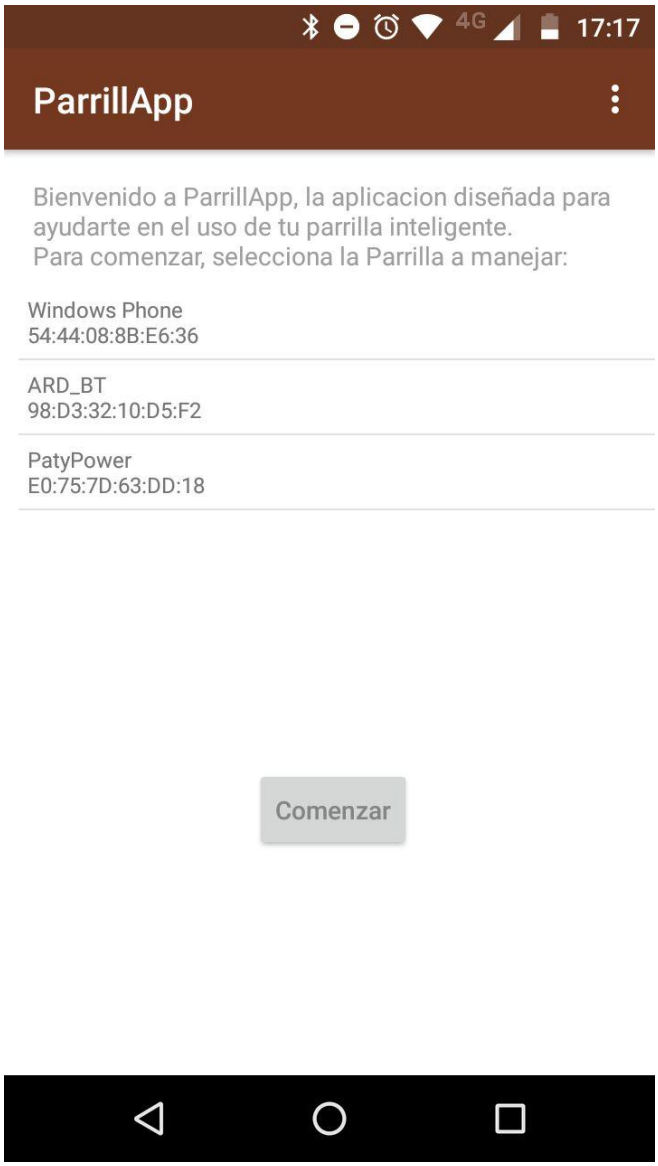
Caso 1: Conexión bluetooth	
	Vinculación con el teléfono OK
Caso 2: Indica temperatura máxima y mínima y se va de la app	
	Entra a modo automático, utilizando temp max y temp min ingresadas
Caso 3: Ingreso altura máxima y mínima	
	Setea los parámetros de Tope máximo y tope mínimo (para cuando mueva la parrilla)
Caso 4: Temperatura actual mayor a Temp máxima ingresada	
	Con el Shake: Sube la parrilla hasta la altura ingresada desde la app
Caso 5: Temperatura actual menor a temp. mínima ingresada	
	Con el Shake: Baja la parrilla hasta la altura ingresada desde la app
Caso 6: Mientras detecta exceso o disminución de temperatura se detecta lluvia	
	Enciende LED azul
	Notificación al teléfono desde la aplicación
Caso 7: Buzzer encendido x proximidad- Apagar desde la aplicación	
	Buzzer apagado
Caso 8: Led encendido + Apagar LED desde la app	
	Led apagado
Caso 9: Envió apagar LED desde el modo manual, luego ingreso al modo automático	
	Led apagado sin volver a encenderse
	Retoma funciones de modo automático

Prototipo Terminado

Sistema Embebido



Aplicación Android


17:17
ParrillApp
Bienvenido a ParrillApp, la aplicacion diseñada para ayudarte en el uso de tu parrilla inteligente.
Para comenzar, selecciona la Parrilla a manejar:
Windows Phone
54:44:08:8B:E6:36
ARD_BT
98:D3:32:10:D5:F2
PatyPower
E0:75:7D:63:DD:18
Comenzar


7:00
ParrillApp
15
60
(TEMPERATURA MINIMA - TEMPERATURA MAXIMA)
ENVIAR
Bajo
Medio
Alto
(NIVEL DE LUZ AMBIENTE)
2
(CENTIMETROS A MOVER)
Estado Actual:
Estado Temperatura: -
Altura Actual: -
Temperatura: -
Lluvia Detectada: -
Proximidad: -
LED: -
Buzzer: -