# MANUAL TÉCNICO:
# PIQUEEM
## Aplicación móvil - Sensor de temperatura

## Software y hardware.

**Sebastián Franco**
**Santiago Londoño**
**Daniel Velasquez**
**Juan Pablo Narvaez**

**Electrónica digital**
**Universidad tecnológica de Pereira**
**Pereira, Julio del 2019**

## Capítulo 1: Introducción.

Piqueem es el resultado de una actividad académica propuesta en la materia electrónica digital dictada por el docente Ramiro Ramírez en la universidad tecnológica de Pereira en el semestre académico 2019-1. Es el conjunto de software y hardware que busca satisfacer problemas relacionados a las temperaturas dentro de un hogar y este apartado escrito busca dar a entender a mayor profundidad el funcionamiento interno del proyecto desde el código de la aplicación móvil hasta las conexiones físicas del arduino.

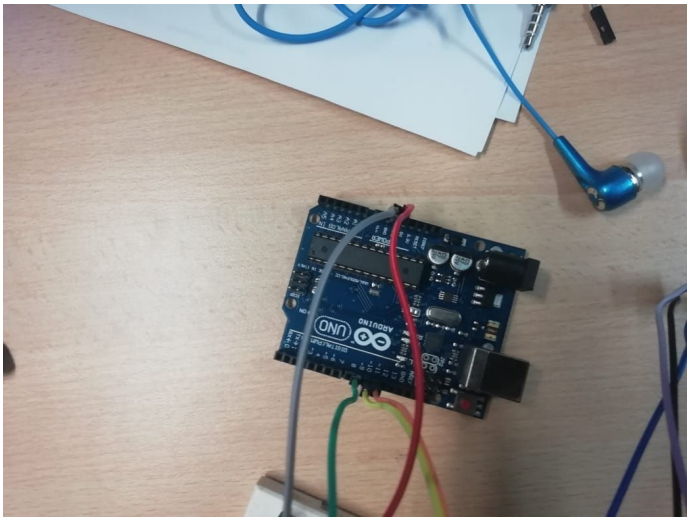
En el capítulo 2 se interioriza el funcionamiento del hardware mientras que en el capítulo 3 se entiende el código de la aplicación móvil desarrollado en android studio. Finalmente en el capítulo 4 se expone el estado actual del proyecto y hacia donde tiende a dirigirse si llega a convertirse en algo factible dentro de la realidad.

## Capítulo 2: Hardware.

El desarrollo del hardware dentro del proyecto está planteado en Arduino, donde se a continuación se exponen sus conexiones físicas con el sensor de temperatura DS18B20 y posteriormente el código que le permite leer temperaturas del ambiente.

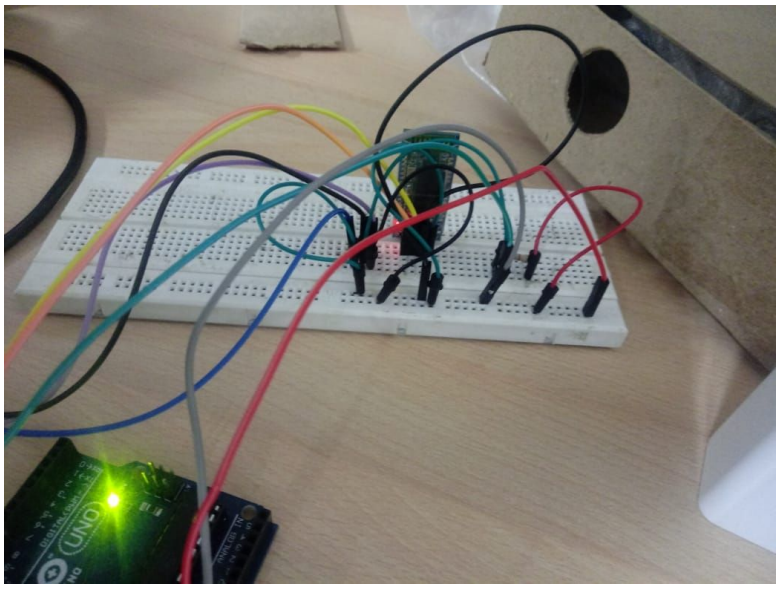
### 2.1: Conexiones físicas.

Las conexiones de hardware se implementan entre la tarjeta de desarrollo Arduino, una protoboard y el sensor de temperatura DS18B20.



Se introduce la placa de desarrollo Arduino utilizada. Tiene un total de 5 conexiones, donde las 2 superiores implican la alimentación eléctrica y las tres de abajo se encargan de enviar los datos del sensor de temperatura, una conexión a la

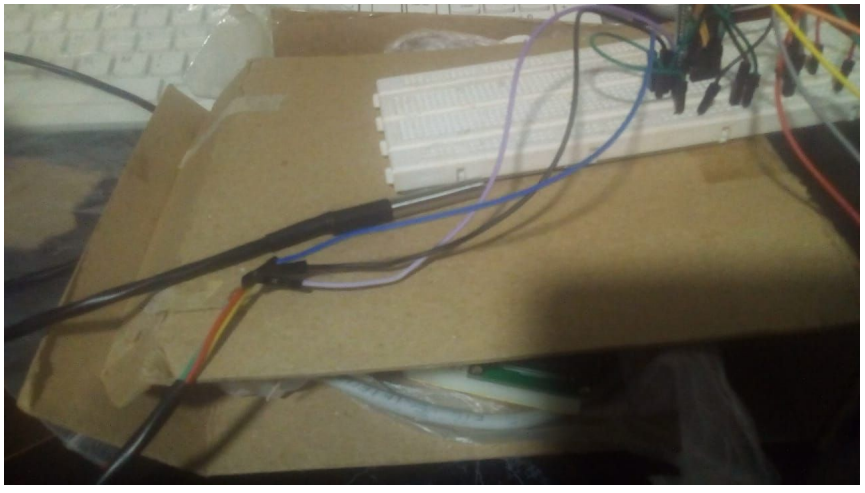
protoboard para asegurar la transmisión de los datos y la conexión a tierra respectivamente.



La fila más cercana a la tarjeta de desarrollo Arduino se encarga de dar electricidad a todo el sistema, es decir, a la tarjeta, el sensor y el módulo bluetooth.

En la siguiente fila superior se tienen las conexiones relacionadas con el GND, es decir, el transporte de datos y la conexión a tierra.

Finalmente se tiene la conexión al módulo bluetooth más arriba el cual se sincroniza con el código interno del Arduino.



Finalmente se tiene el sensor de temperatura DS18B20.el cual tiene un total de tres conexiones con la protoboard y una con la tarjeta Arduino. Las conexiones con la protoboard son: Conexión al GND para transporte de datos, conexión a tierra y conexión con el módulo bluetooth.

## 2.2: Código interno del Arduino.

Dentro de la placa de desarrollo Arduino se debe sintetizar la misma para hacerla apta de recibir datos del sensor de temperatura. El código que permite la lectura se anexa de la siguiente manera:

```
void loop()
{

  if(BT.available())    // Si llega un dato por el puerto BT se envía al monitor serial
  {
    char VarChar;
    VarChar = BT.read();
    if (VarChar == '1')
    {
      char Aux='1';
      while(Aux=='1'){
        delay(100);
        sensors.requestTemperatures();   //Se envía el comando para leer la temperatura
        float temp= sensors.getTempCByIndex(0); //Se obtiene la temperatura en °C
        BT.print(String(temp));
        BT.print("#");
        char Aux2;
        Aux2 = BT.read();
        if (Aux2=='0'){
          Aux=0;
        }
      }
    }
  }

}
```

# Capítulo 3: Software

El software implementado en este proyecto se representa en una aplicación móvil (para detalles gráficos vea manual de usuario) desarrollada en Android studio en el tiempo de 3 semanas. A continuación se muestran capturas de pantalla relacionadas a los funcionamientos de distintas zonas de la aplicación móvil.

Los módulos de la aplicación funcionan bajo el mismo código excepto el módulo personalizado el cual tiene sus propias capturas de pantalla.

## 3.1: Funciones de la app móvil

```
//Comprueba que el dispositivo Bluetooth Bluetooth está disponible y solicita que se active si está desactivado
private void VerificarEstadoBT() {

    if(btAdapter==null) {
        Toast.makeText(getBaseContext(), text: "El dispositivo no soporta bluetooth", Toast.LENGTH_LONG).show();
    } else {
        if (btAdapter.isEnabled()) {
        } else {
            Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBtIntent, requestCode: 1);
        }
    }
}
```

Comprueba si el dispositivo Bluetooth está disponible y solicita que se active si está desactivado.

```
    public void run()
    {
        byte[] buffer = new byte[256];
        int bytes;

        // Se mantiene en modo escucha para determinar el ingreso de datos
        while (true) {
            try {
                bytes = mmInStream.read(buffer);
                String readMessage = new String(buffer, offset: 0, bytes);
                // Envia los datos obtenidos hacia el evento via handler
                bluetoothIn.obtainMessage(handlerState, bytes, arg2: -1, readMessage).sendToTarget();
            } catch (IOException e) {
                break;
            }
        }
    }
    //Envio de trama
    public void write(String input)
    {
        try {
            mmOutStream.write(input.getBytes());
        }
        catch (IOException e)
        {
            //si no es posible enviar datos se cierra la conexión
            Toast.makeText(getBaseContext(), text: "La Conexión fallo", Toast.LENGTH_LONG).show();
            finish();
        }
    }
}
```

Recepción de datos (método run) y el envío de datos (método write, se envían solo dos datos: 0 y 1 cuando se presionan los botones encendido y apagado)

```
private BluetoothSocket createBluetoothSocket(BluetoothDevice device) throws IOException
{
    //crea un conexion de salida segura para el dispositivo
    //usando el servicio UUID
    return device.createRfcommSocketToServiceRecord(BTMODULEUUID);
}
```

Conexión de salida.

```
bluetoothIn = new Handler() {
    public void handleMessage(android.os.Message msg) {
        if (msg.what == handlerState) {
            String readMessage = (String) msg.obj;
            DataStringIN.append(readMessage);

            int endOfLineIndex = DataStringIN.indexOf("#");

            if (endOfLineIndex > 0) {
                String dataInPrint = DataStringIN.substring(0, endOfLineIndex);
                IdBufferIn.setText(dataInPrint);//<-<- PARTE A MODIFICAR >->->
                temperatura = Float.parseFloat(dataInPrint);
                switch (funcion){
                    case 1:
                        if (temperatura>=32){
                            addNotification( title: "Temperatura biberón adecuada", dataInPrint);
                        }
                        else if(temperatura>=35){
                            addNotification( title: "Temperatura biberón crítica", dataInPrint);
                        }
                        break;
                    case 2:
                        if (temperatura>=35){
                            addNotification( title: "Temperatura agua adecuada", dataInPrint);
                        }
                        else if(temperatura>=37){
                            addNotification( title: "Temperatura agua crítica", dataInPrint);
                        }
                        break;
                    case 3:
                        if (temperatura>=100){
                            addNotification( title: "Temperatura crítica del aceite", dataInPrint);
                        }
                        break;
                    case 4:
                        if (temperatura>=60 || temperatura<=4){
                            addNotification( title: "Temperatura comida", dataInPrint);
                        }
```

Lista de casos que se encarga de dar las alarmas respectivas de acuerdo a la opción que se de (biberones, agua, aceite, comida)

```
private void addNotification(String title, String data) {
    // Construir la notificacion
    NotificationCompat.Builder builder = new NotificationCompat.Builder( context: this)
            .setSmallIcon(R.mipmap.ic_launcher_round)
            .setContentTitle(title)
            .setVibrate(new long[] {1000, 1000})
            .setSound(Settings.System.DEFAULT_NOTIFICATION_URI)
            .setContentText("Temperatura: " + data);

    // Crear el Intent para la notificacion
    Intent notificationIntent = new Intent( packageContext: this, MainActivity.class);
    PendingIntent contentIntent = PendingIntent.getActivity( context: this,  requestCode: 0, notificationIntent, PendingIntent.FLAG_UPDATE_CURRENT);
    builder.setContentIntent(contentIntent);


    // Agregar como notificacion
    NotificationManager manager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
    manager.notify( id: 0, builder.build());
}
```

Método que utilizan todas las ventanas de opciones para poner la notificación

```
public class PersonalizarActiv extends AppCompatActivity {
    float temperatura=28; //Temperatura promedio
    float MaximaTemp = Float.POSITIVE_INFINITY, MinimaTemp = Float.NEGATIVE_INFINITY;
```

Variables float para los condicionales de la temperatura de la pantalla personalizable

```
bluetoothIn = new Handler() {
    public void handleMessage(android.os.Message msg) {
        if (msg.what == handlerState) {
            String readMessage = (String) msg.obj;
            DataStringIN.append(readMessage);

            int endOfLineIndex = DataStringIN.indexOf("#");

            if (endOfLineIndex > 0) {
                String dataInPrint = DataStringIN.substring(0, endOfLineIndex);
                IdBufferIn.setText(dataInPrint);//<-<- PARTE A MODIFICAR >->->
                temperatura = Float.parseFloat(dataInPrint);
                if (temperatura>=MaximaTemp){
                    addNotification( title: "Temperatura Maxima alcanzada", dataInPrint);
                }
                else if(temperatura<=MinimaTemp){
                    addNotification( title: "Temperatura Mínima alcanzada", dataInPrint);
                }
                DataStringIN.delete(0, DataStringIN.length());
            }
        }
    }
};
```

Notificación de la pantalla personalizada

```
IdEncender.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v)
        {
            IdLetrero.setText("Encendido");
            MyConexionBT.write( input: "1");
        }
    });

IdApagar.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        IdLetrero.setText("Apagado");
        MyConexionBT.write( input: "0");
    }
});

IdDesconectar.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        if (btSocket!=null)
        {
            try {btSocket.close();}
            catch (IOException e)
            { Toast.makeText(getBaseContext(), text: "Error", Toast.LENGTH_SHORT).show();;}
        }
        finish();
    }
});
}
```

Métodos para los botones de todas las opciones menos la personalizable

## Capítulo 4: Utópicas implementaciones y aclaraciones.

El desarrollo del proyecto se encuentra en una fase académica y su tiempo e intensidad de desarrollo se asimilan a lo que representa. El trabajo no se encuentra explícitamente pulido ni posee un apartado gráfico de nivel comercial, sin embargo, en un utópico caso de progreso de proyecto se propone mejorar de manera

significativa el apartado gráfico mientras la sensibilidad del sensor se mejora para que tenga reacciones más rápidas y fieles a las necesidades de un mundo real.

Para mayor información, quejas o solicitudes contactar con sebastian.franco@utp.edu.co

Para revisar el repositorio de este proyecto, ingresar en:
https://github.com/Andriuk/SensorDeTemperatura