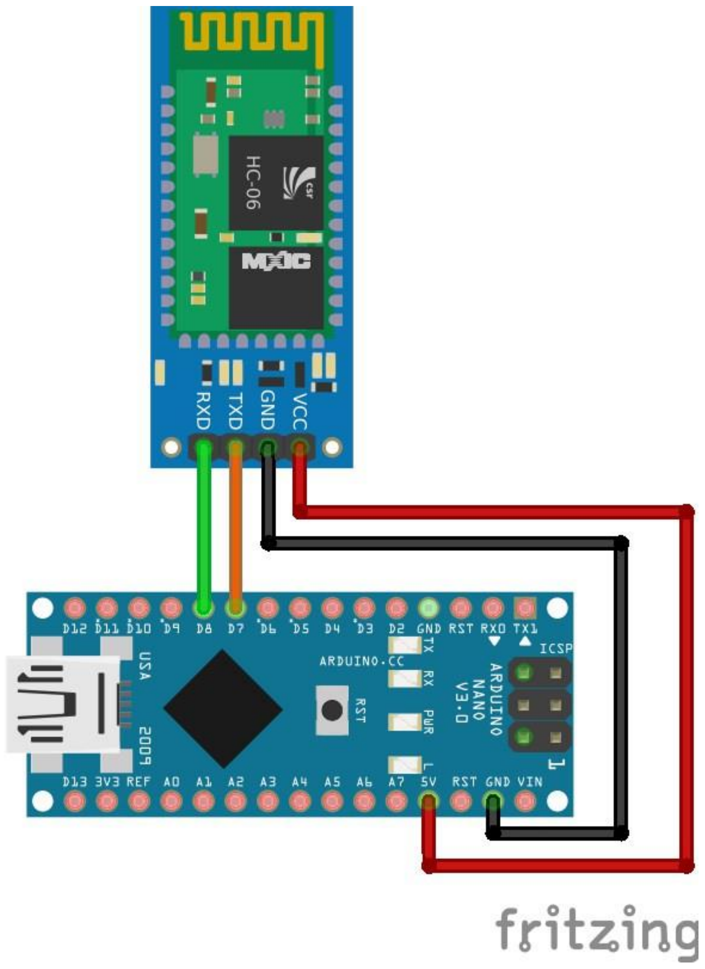
Mini tutorial for the Hardware team

How to work with HC-05 module with arduino

Step 1: Preparing HC-05 and Arduino


HC-06
RXD
TXD
GND
VCC
D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN
ARDUINO.CC
ARDUINO NANO V3.0
ICSP
fritzing

Step 2: Programing the circuit

```
#include <SoftwareSerial.h> // Include this library to work with the HC-05 Bluetooth
module

SoftwareSerial BTserial(10, 11); // RX | TX

int sensorPin = A0; //Sensor value input for example a potentiometer

int sensorValue = 0;

void setup() {

       BTserial.begin(9600);
}

void loop() {

       sensorValue = analogRead(sensorPin); //Reading the sensor value

       BTserial.print(sensorValue); //Sending the sensor value via Bluetooth

       delay(20);

}
```

Check this tutorial for more info to how to send data via Bluetooth to an android smartphone: http://www.instructables.com/id/How-to-Receive-Arduino-Sensor-Data-on-Your-Android/

Working with JSON data in Arduino and sending it via Bluetooth

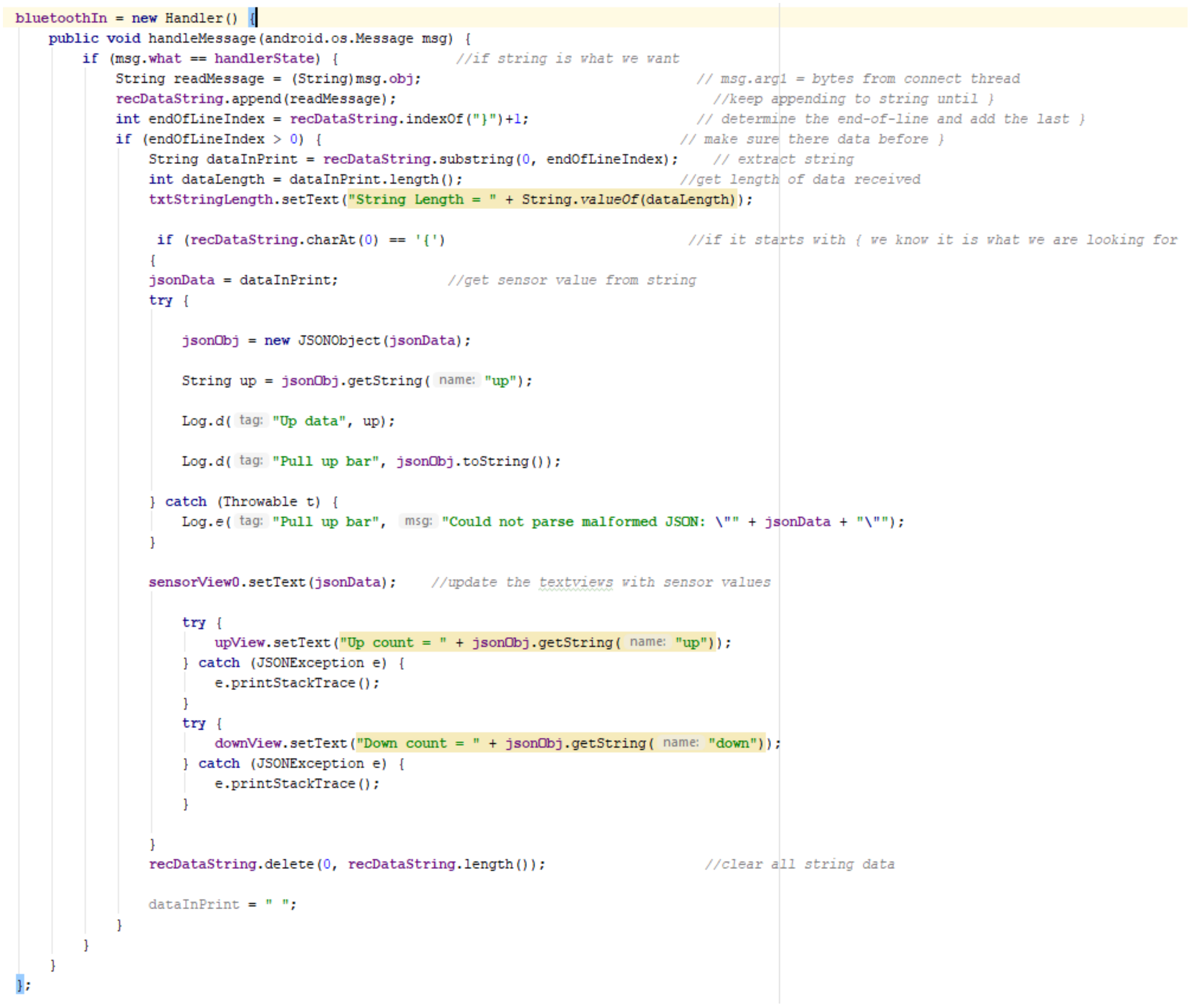
I add two buttons to the circuit to simulate de up and down when you exercise in the pull up bar and another to reset the values. Below is the entire code that I used in the circuit to send JSON data.

```
#include <ArduinoJson.h> // Include this library to work with JSON data but you have
to install it first. See for more info https://arduinojson.org/
#include <SoftwareSerial.h> // Include this library to work with the HC-05 Bluetooth
module


// Memory pool for JSON object tree.
//
// Inside the brackets, bufferSize is the size of the pool in bytes.
// Don't forget to change this value to match your JSON document.
// Use arduinojson.org/assistant to compute the capacity.
//*StaticJsonBuffer<200> jsonBuffer;

// StaticJsonBuffer allocates memory on the stack, it can be
// replaced by DynamicJsonBuffer which allocates in the heap.
//
const size_t bufferSize = JSON_OBJECT_SIZE(3);
DynamicJsonBuffer jsonBuffer(bufferSize);

// Create the root of the object tree.
//
// It's a reference to the JsonObject, the actual bytes are inside the
// JsonBuffer with all the other nodes of the object tree.
// Memory is freed when jsonBuffer goes out of scope.
JsonObject& root = jsonBuffer.createObject();

SoftwareSerial BTserial(10, 11); // RX | TX

const int upPin = 2; //Button for the up counting
const int downPin = 3; //Button for the down counting
const int resetPin = 4; //Reset button

int downState = 0;
int upState = 0;
int resetState = 0;

//To store the millis
//
long pullUp = 0;
long pullDown = 0;


void setup() {

       pinMode(upPin, INPUT);
       pinMode(downPin, INPUT);
       pinMode(resetPin, INPUT);

       //Initialize Serial Bluetooth
       BTserial.begin(9600);

       // Initialize Serial port
       Serial.begin(9600);
       while (!Serial) continue;
```

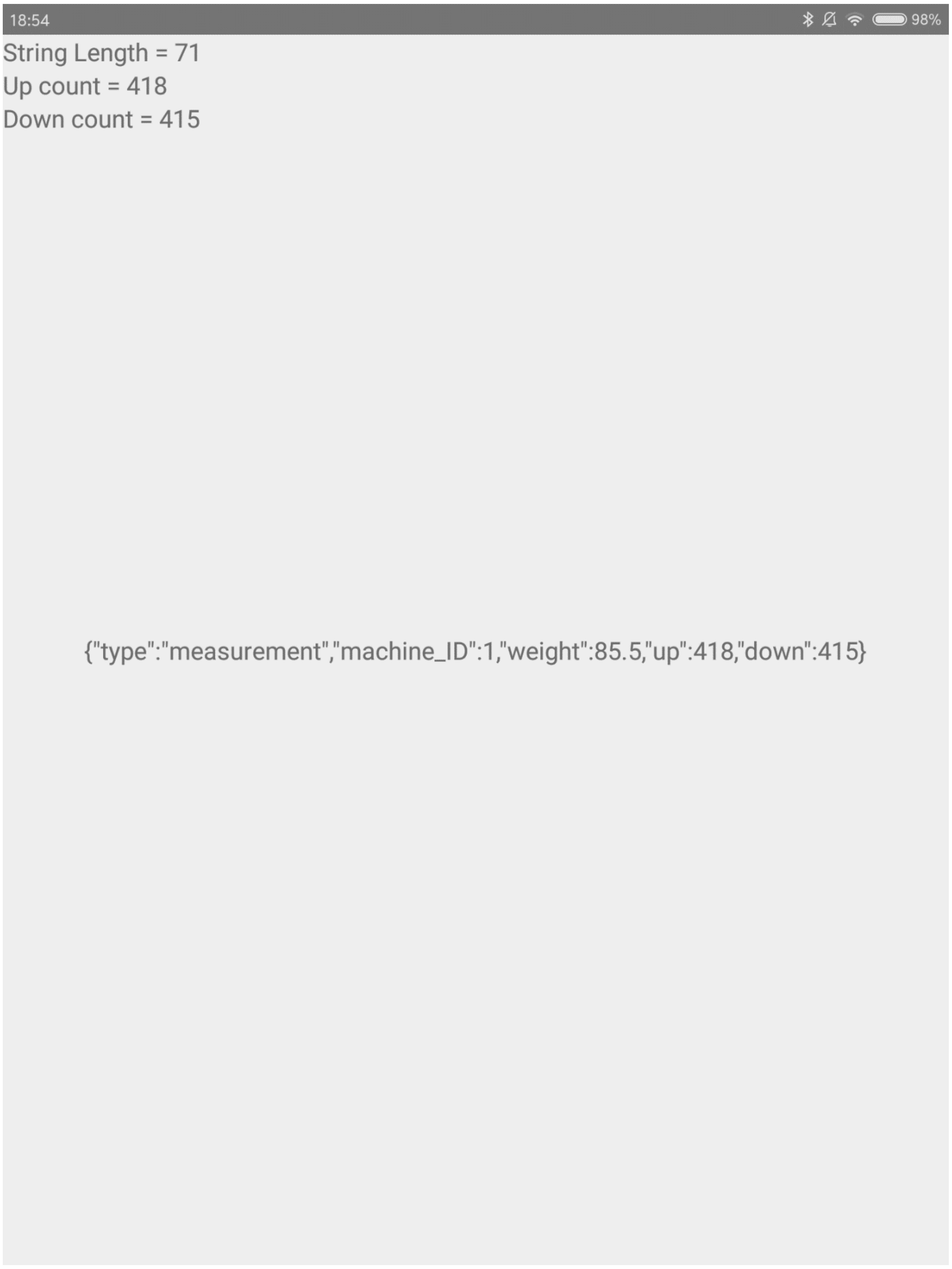
```
	// Add values in the object
	//
	// Most of the time, you can rely on the implicit casts.
	// In other case, you can do root.set<long>("time", 1351824120);


	// Add a nested array.
	//
	// It's also possible to create the array separately and add it to the
	// JsonObject but it's less efficient.
	//JsonArray& data = root.createNestedArray("data");
	//data.add(48.756080);
	//data.add(2.302038);

	root.printTo(Serial);

	// This prints for example JSON in 1 line:
	// {"sensor":"gps","time":1351824120,"data":[48.756080,2.302038]}

	Serial.println();

	root.prettyPrintTo(Serial);

	// This prints for exapmle:
	// {
	//   "sensor": "gps",
	//   "time": 1351824120,
	//   "data": [
	//     48.756080,
	//     2.302038
	//   ]
	// }
}

void loop() {

	//Simulating the data with 2 buttons

	unsigned long currentMillis = millis(); //Starting time to count the up and down
in the pull up bar like in the dummy date

	//Reading buttons
	//
	downState = digitalRead(downPin);
	upState = digitalRead(upPin);
	resetState = digitalRead(resetPin);

	if (resetState == HIGH) {
		pullUp = 0;
		pullDown = 0;
		currentMillis = 0;
	}

	if (upState == HIGH) {
		pullUp = currentMillis;
	}
	else if (downState == HIGH) {
		pullDown = currentMillis;
	}
```

```
        //Adding values to the JSON structure
        //
        if (pullUp > 0 | pullDown > 0) {
                root["type"] = "measurement";
                root["up"] = pullUp;
                root["down"] = pullDown;
        }

        //Initial state before pressing any button
        //
        else {
                root["type"] = "Initial";
                root["machine_ID"] = 1;
                root["weight"] = 85.5;
        }

        //Printing JSON structure in a string
        //
        String output;
        root.printTo(output);

        //Sending the JSON data in a string via BLuetooth
        BTserial.print(output);

        delay(20);
}
```

Receiving the JSON string from the Arduino and making it a JSON object



```java
bluetoothIn = new Handler() {
    public void handleMessage(android.os.Message msg) {
        if (msg.what == handlerState) {                 //if string is what we want
            String readMessage = (String)msg.obj;                             // msg.arg1 = bytes from connect thread
            recDataString.append(readMessage);                                  //keep appending to string until }
            int endOfLineIndex = recDataString.indexOf("}")+1;                // determine the end-of-line and add the last }
            if (endOfLineIndex > 0) {                                       // make sure there data before }
                String dataInPrint = recDataString.substring(0, endOfLineIndex);    // extract string
                int dataLength = dataInPrint.length();                      //get length of data received
                txtStringLength.setText("String Length = " + String.valueOf(dataLength));

                 if (recDataString.charAt(0) == '{')                         //if it starts with { we know it is what we are looking for
                {
                jsonData = dataInPrint;             //get sensor value from string
                try {

                    jsonObj = new JSONObject(jsonData);

                    String up = jsonObj.getString( name: "up");

                    Log.d( tag: "Up data", up);

                    Log.d( tag: "Pull up bar", jsonObj.toString());

                } catch (Throwable t) {
                    Log.e( tag: "Pull up bar",  msg: "Could not parse malformed JSON: \"" + jsonData + "\"");
                }

                sensorView0.setText(jsonData);    //update the textviews with sensor values

                    try {
                        upView.setText("Up count = " + jsonObj.getString( name: "up"));
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                    try {
                        downView.setText("Down count = " + jsonObj.getString( name: "down"));
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }

                }
                recDataString.delete(0, recDataString.length());                   //clear all string data

                dataInPrint = " ";
            }
        }
    }
};
```



```
18:54                                                98%
String Length = 71
Up count = 418
Down count = 415
```

{"type":"measurement","machine_ID":1,"weight":85.5,"up":418,"down":415}

* All the source code in this document is in our github repository !