USING BLUETOOTH'S TO CREATE A SENSOR NETWORK

Sam Winchcombe
Sam.winchcombe@hotmail.com

Contents

Multi-sensor Arduino Bluetooth temperature sensor

Introduction

This project explains the method I used to transmit the temperature readings from multiple sensor modules to a master module using Bluetooth and aims to help you recreate the same system.

Bluetooth Overview

The Bluetooth modules were HM-10's operating Bluetooth V4.0, Arduino Uno's were used as MCUs and TMP36's were used for the temperature sensors.

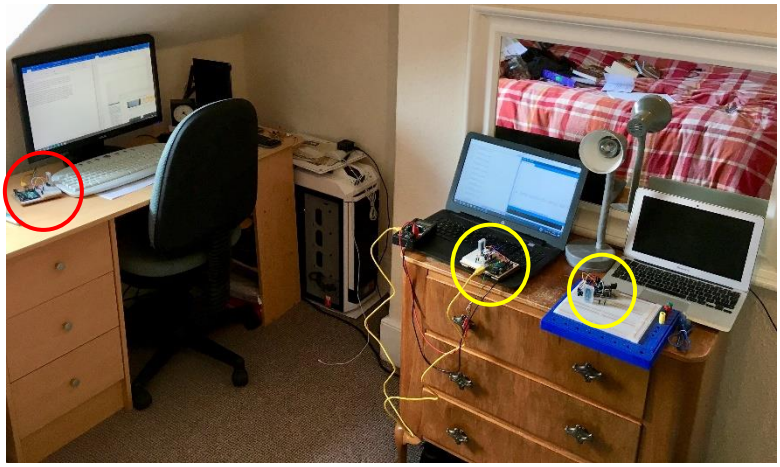


Figure 1: Experimental set up of Master module (circled in Red) and the sensor modules (circled in yellow)

In theory Bluetooth has two network modes, although newer versions have a third. Bluetooth modules can either operate as a piconet or in advertising mode. Piconets are the most common format talked about, however I have yet to see any real-world products use them and couldn't recreate one for myself. In theory a Piconet consists of up to 8 slaves connecting to one master which requests information from the slaves, as seen in figure 2. Advertising mode, can itself be used in two modes. It can be used to broadcast information in a non-connectable mode or used to advertise the availability of a module for connection, see figure 3. It's this non connectable mode that I used to create this wireless network which is explained in this project.

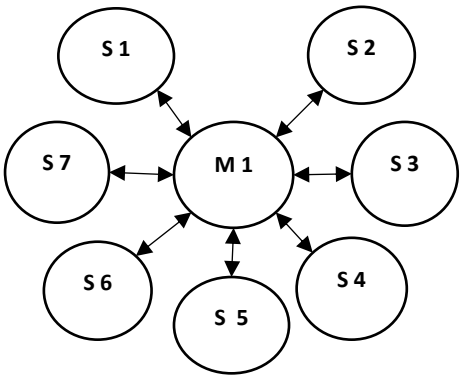


Figure 2 Piconet Diagram

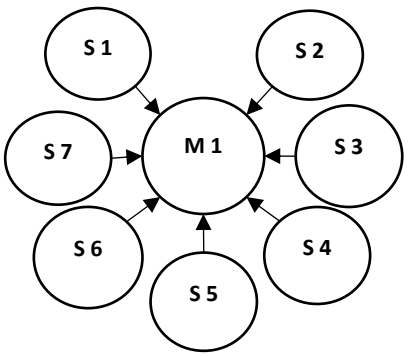


Figure 3 BLE Advertising Mode

Advertising mode and payload data format

The advertising structure used depends on the what structure is being used in the Bluetooth module. It is possible to create your own, but this was outside of the scoop of the project. The HM-10 uses the iBeacon structure, which was designed to ping shoppers' phones with deals as they walked past shops.

The iBeacon data structure consists of 31Bytes in the payload and the HM-10 follows the following structure.

Advertisement packet structure:

Company ID : UUID : Major, Minor & RSSI : MAC Address : RSSI (dBm)

4C000215 : 74278BDAB64445208F0C720EAF059935 : 1234FA01C5 : D8A98BB13644 : -071

Table 1: Advertising Data Structure Breakdown

Company ID	4C000215
UUID (Unique User ID)	74278BDAB64445208F0C720EAF059935
Major	1234
Minor	FA01
RSSI	C5
MAC ADDRESS	D8A98BB13644
RSSI (dBm) – measured at 1m for comparison	-071

Note: I have seen it shown differently elsewhere.

Setting the UUID's of a HM-10

To set the UUID's I used the 'Talking_to_hm-10' Script from Martyn Currey's website under the subheading "Getting an Arduino talking to the HM-10". This requires the AltSoft Serial Library to communicate to the HM-10 module from the Arduino Uno. The script needs to be uploaded onto the Arduino and the Serial Monitor to communicate with the HM-10 as shown in figure 7 on pages 6 & 7. Using the wiring diagram in figure 6 and following the AT commands under the Subheading "HM-10 as an iBeacon" you will be able to change the UUID.

Note: Careful of Baud rate and line ending setting. I sometimes had to enter the initial AT command twice to get a response. More AT commands are available on Martyn Currey's website.

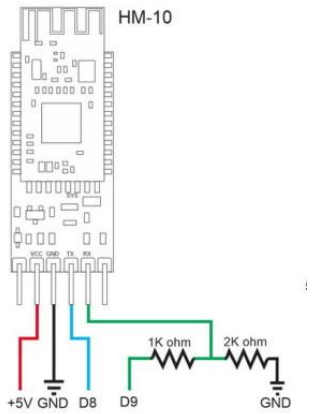
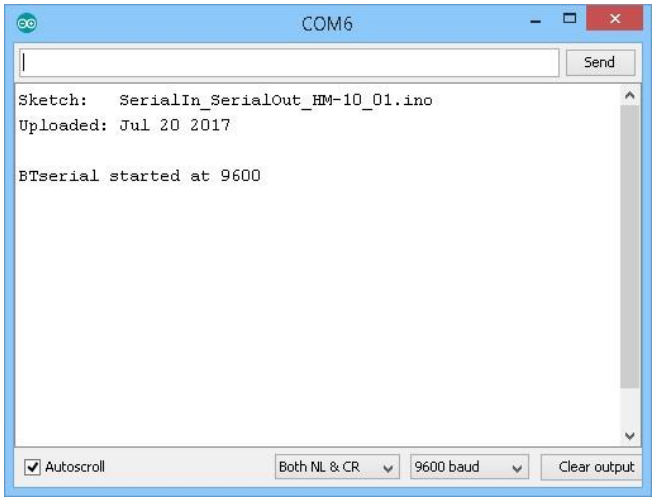
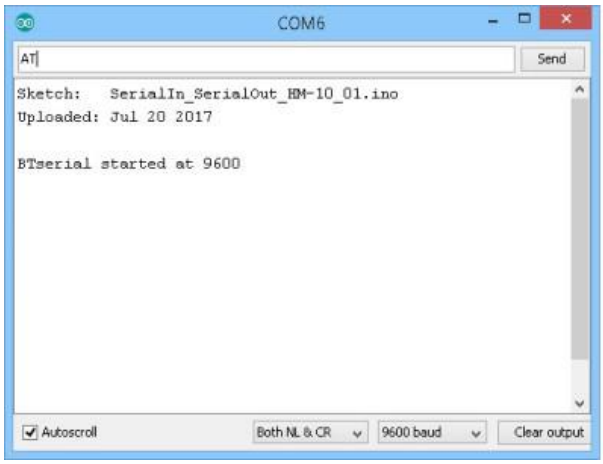
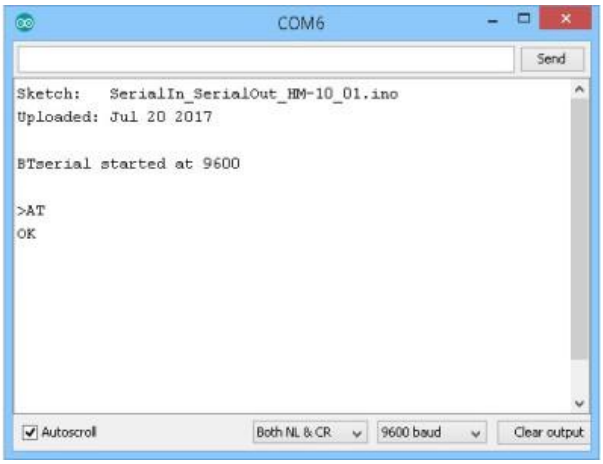
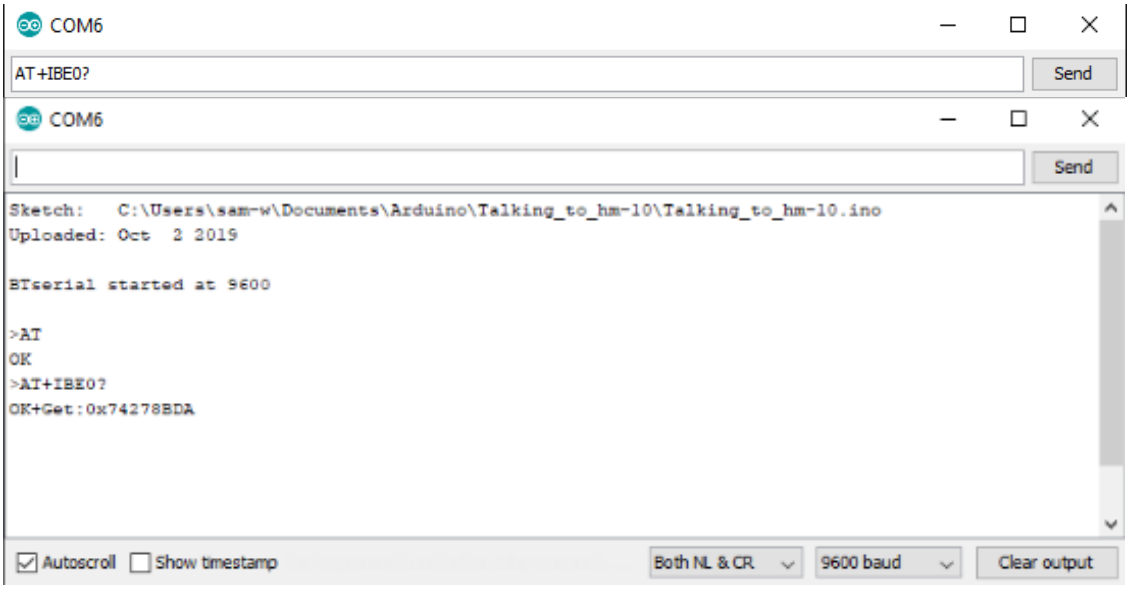


Figure 6 : Wiring for Hm-10 module for UUID configuration. View from back


COM6
Send
Sketch: SerialIn_SerialOut_HM-10_01.ino
Uploaded: Jul 20 2017
BTserial started at 9600
Autoscroll
Both NL & CR
9600 baud
Clear output


COM6
AT
Send
Sketch: SerialIn_SerialOut_HM-10_01.ino
Uploaded: Jul 20 2017
BTserial started at 9600
Autoscroll
Both NL & CR
9600 baud
Clear output


COM6
Send
Sketch: SerialIn_SerialOut_HM-10_01.ino
Uploaded: Jul 20 2017
BTserial started at 9600
>AT
OK
Autoscroll
Both NL & CR
9600 baud
Clear output


COM6
AT+IBE0?
Send
COM6
Send
Sketch: C:\Users\sam-w\Documents\Arduino\Talking_to_hm-10\Talking_to_hm-10.ino
Uploaded: Oct 2 2019
BTserial started at 9600
>AT
OK
>AT+IBE0?
OK+Get:0x74278BDA
Autoscroll
Show timestamp
Both NL & CR
9600 baud
Clear output

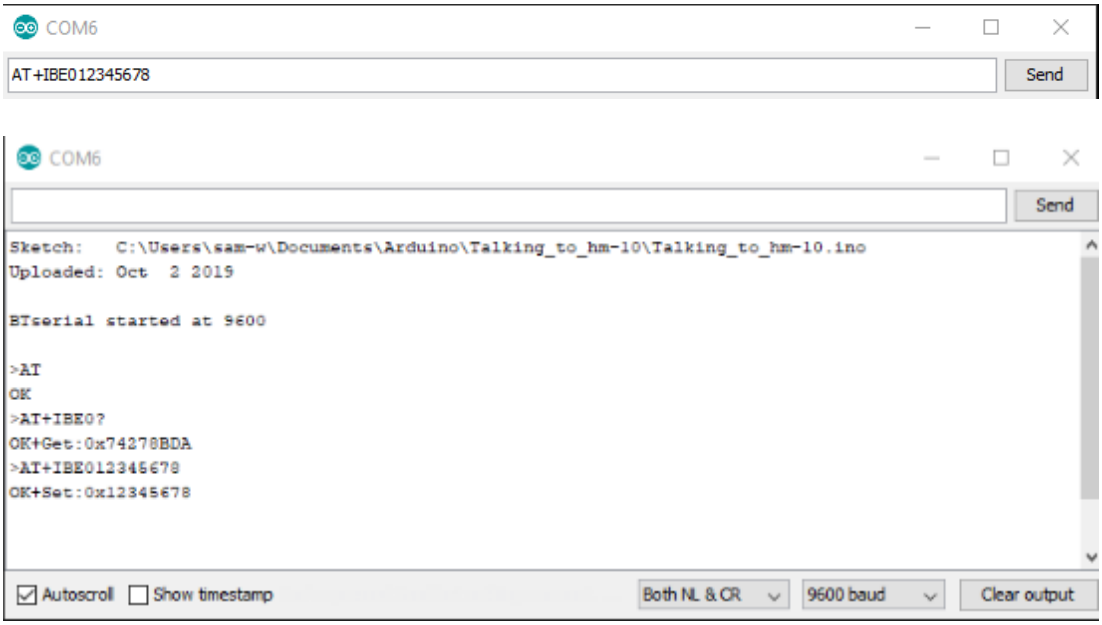


Figure 7: At commands and the responses to configure the UUID.

How the Code Works

The code for the Sensors (previous to this described as slaves) and master modules follow the flow diagram in figure 4 and 5 respectively. Figure 5 can be repeated for 'n' number of sensors. Setting a sensors' UUID so that it's unique is a prerequisite step. See the chapter on "Setting the UUID's of a HM-10".

NOTE: The HM-10 expects the AT commands in ASCII so hex values from 'A->F' values needs to be converted to ASCII.

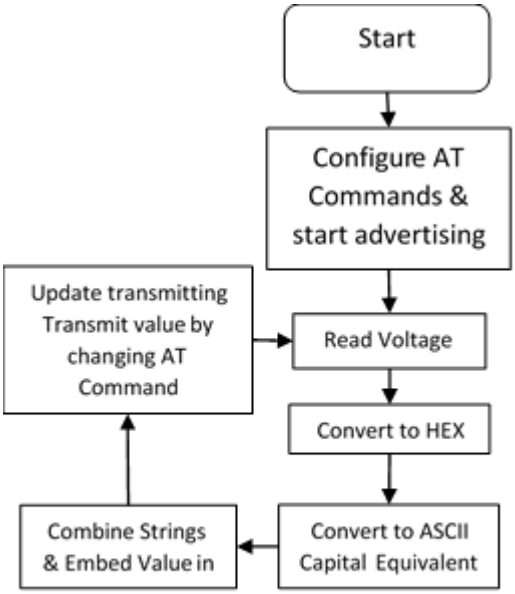


Figure 4: Flow diagram of Slave module

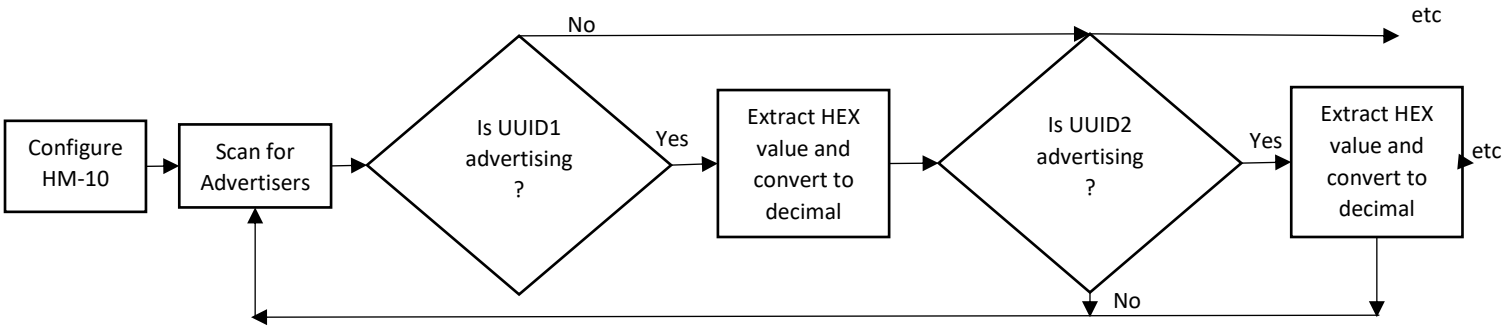


Figure 5: Flow Diagram of Master Module

Project set up – adjustments to code and wiring diagrams

The masters and slave modules need to be wired as seen in figures 8 and 9 respectively.

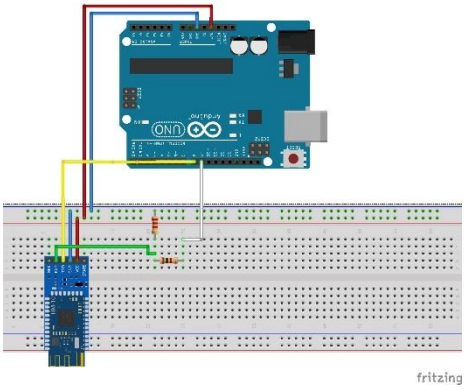


Figure 8: Wiring for master module

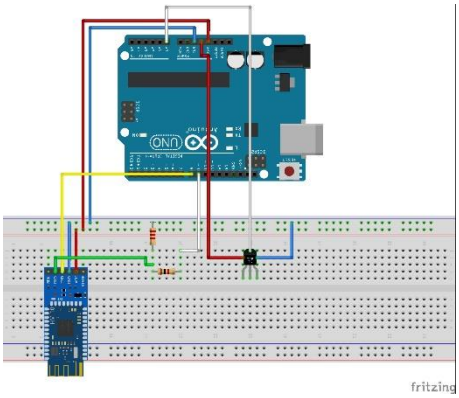


Figure 9: Wiring for temperature sensing module

In the master modules' code the UUID's of the sensors needs to be changed to match those of the HM-10s on the senor modules.The relevant code is on line 20.

Note: The code for the Sensor modules is heavily based off of the code from this project by Raymond Genovese.

Expected Results

When both modules are set up and running the master should output the advertisers its detecting and then extract the temperature value from each sensor. You can comment out these lines and use the output however you wish. An example of this can be seen in figure 10. Other advertising Bluetooth devices will be picked up but as they don't have the UUIDs that the master is looking for they will be ignored.

```
20:08:18.577->OK+DISISOK+DISC:00000000:00000000000000000000000000000000:0000000000:00989A113F4D:-069
              OK+DISC:00000000:00000000000000000000000000000000:0000000000:11EBD9F4B2D8:-066
              OK+DISC:00000000:00000000000000000000000000000000:0000000000:6ACA3DA98E6F:-092
              OK+DISC:4C000215:74278BDAB64445208F0C720EAF059935:123400A2C5:D8A98BB13644:-074
              OK+DISC:00000000:00000000000000000000000000000000:0000000000:489F46CE8370:-058
              OK+DISC:00000000:00000000000000000000000000000000:0000000000:7DFBC4799A75:-080
              OK+DISC:4C000215:12345678B64445208F0C720EAF059935:1234008DC5:D8A98BB0D073:-091
              OK+DISCE
20:08:19.140 -> Sensor 1:
20:08:19.206 -> Temperature: 29.10 Degrees Celsius
20:08:19.272 -> Sensor1 found
20:08:19.272 -> Sensor 2:
20:08:19.340 -> Temperature: 18.85 Degrees Celsius
20:08:19.373 -> Sensor2 found
20:08:23.348->OK+DISISOK+DISC:00000000:00000000000000000000000000000000:0000000000:11EBD9F4B2D8:063
              OK+DISC:4C000215:74278BDAB64445208F0C720EAF059935:123400A2C5:D8A98BB13644:081
              OK+DISC:00000000:00000000000000000000000000000000:0000000000:00989A113F4D:071
              OK+DISC:00000000:00000000000000000000000000000000:0000000000:7DFBC4799A75:087
              OK+DISC:00000000:00000000000000000000000000000000:0000000000:6ACA3DA98E6F:091
              OK+DISC:00000000:00000000000000000000000000000000:0000000000:489F46CE8370:-066
              OK+DISCE
20:08:23.849 -> Sensor1:
20:08:23.915 -> Temperature: 29.10 Degrees Celsius
20:08:23.948 -> Sensor1 found
20:08:23.948 -> Sensor2 not found
```

Figure 10 : An example of the expected response

There are multiple ways in which this code can be improved from improving the efficiency of the power consumption to allowing a wider range of temperature values to be discovered. However, this was outside of the scoop of this project as at the time I was only trying to prove of concept.

Sources:

http://www.martyncurrey.com/hm-10-bluetooth-4ble-modules/

https://www.analog.com/media/en/technical-documentation/data-sheets/TMP35_36_37.pdf

https://www.allaboutcircuits.com/projects/build-an-arduino-uno-multi-node-ble-humidity-and-temperature-sensor-monitor/