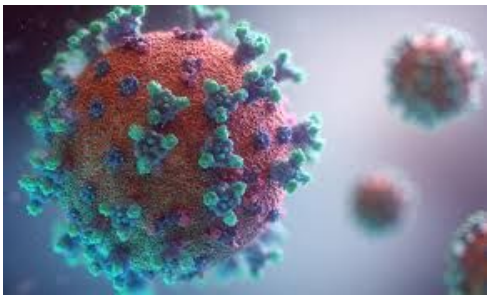
Contactless Body Temperature Scanner and Sensing System



Final report

Ryan Dobbe, Colton Martin, and Briana Parker

Department of Computer Science
Sam Houston State University

Date of Completion: 12/10/2020

Table of Contents

1 Executive summary

The year 2020 has proven to have many challenges to say the least. The Covid-19 pandemic has made life significantly more difficult for many people. However, one thing is certain, the integration and improvement of technology in our everyday lives has become an essential and necessary support crutch. As we hear about the shortages of certain technology and equipment deemed necessary by public health officials through traditional and social media, it is becoming more apparent that we need both an effective and cheap solution to help combat this virus. Specifically, we need more effective ways to help improve our preventative measures and procedures as well as an effective way of monitoring those that might potentially be infected to reduce possible spread of the virus.

One of the more commonly discussed pieces of equipment that saw massive shortages in the early stages of the virus were the ventilators. The cost of these ventilators can range anywhere from 5000 - 50,000 USD with the cost of some of the more premium ventilators averaging over 25,000 dollars. During those early months of the pandemic, when the ventilator shortages were at their worst, the number of US hospitalizations at the time was roughly 60,000 or more. One can see how the shortage might be problematic when calculating the average cost of every patient needing their own ventilator to be approximately 1.5 billion dollars on the conservative end. This doesn't even include the cost of one patient occupying a hospital bed, especially when there is a shortage of those as well as hospital staff becoming overworked and infected themselves.

When cost is factored into the equation, it becomes increasingly obvious that a cheap and effective solution for monitoring and preventing more surge outbreaks is an invaluable resource. This is where the NXP Mbed board becomes a very attractive potential solution. The price of one board, a bluetooth chip, a temperature sensor, as well as an application board totals under $200 USD. Even with included labor costs of configuring all of the hardware and uploading software to run it, it is still an extremely cheap monitoring solution.

The goal of this project is to produce a contactless body temperature scanner and sensing system using the aforementioned hardware. The mbed board will be the microcontroller used as the brains of the operation in conjunction with the contactless infrared temperature sensor, making scanning an individuals' temperature seamless and germ free. The thermal sensor readings will then be encrypted to ensure data integrity and transferred to either a host PC/laptop and/or mobile phone/device via an RN41/RN42 bluetooth module chip.

The contactless temperature sensor in conjunction with the bluetooth communication and further ability to become an IoT device/system with the use of appropriate hardware makes this design extremely intelligent, especially when factoring the relatively low cost of setting up a proper system within a medical environment/facility. It is proposed that the system will detect an individual's body temperature and alert the mbed system if said temperature rises above the high-temperature threshold that is programmed prior to deployment of the system. Detection of an individual's high temperature will be time stamped and stored into a data logger system that is sent wirelessly via the use of the bluetooth module using an encrypted channel to preserve the

integrity of the data. Once encrypted and time-stamped, the thermal data as well as computed hash values of the encryption will be transferred to the data logger system.

2 Project background

The general scope of the project is to develop, design, and deploy a contactless body temperature scanner to help mitigate both general spread and surge outbreaks of Covid-19. The project aims to integrate certain technological innovations including a body temperature data logger, an alert system, appropriate security measures via the use of an encrypted channel, and wireless communication between devices via the use of the RN41/42 bluetooth chip module.

2.1 Needs statement

As we approach the end of 2020, we can see that the number of Covid cases and hospitalizations due to Covid have skyrocketed both globally and nationally far exceeding the previous initial outbreak back in early March. It is hard to say for certain, especially given that directions of health experts seem to be constantly evolving, where the problem truly lies causing such a rampant outbreak as of late. One thing that every health expert does seem to agree on across the globe is that early signs of Covid infection tend to elevate body temperatures above the normal average. This is especially true in certain countries that rely heavily on public transportation such as train stations that were seen using an infrared thermometer on the public before granting entrance to the station. However, the main drawback with this is that it requires a person to operate the thermometer. This is not an ideal scenario as not only does it put the person operating the thermometer at increased risk of exposure to the virus, it also allows for human error which is never a good thing in any situation let alone a global pandemic. A smarter system with less error and higher precision and accuracy with the ability to log data whilst ensuring security and integrity of personal data would be a much more ideal in not only providing accurate readings to help predictions, but also providing a safer monitoring solution.

2.2 Goal and objectives

The goal is to design a detection system with higher accuracy and efficiency than what is currently being implemented. This would involve the proposed mbed contactless body temperature sensor with bluetooth communication properly configured in a room/hallway/facility where a user operator is unnecessary and will achieve both a higher accuracy and higher precision on temperature readings along with a more secure data logging system by using an encrypted channel to ensure the privacy and integrity of the person being scanned. Not only will this increase the amount of data being collected, but will ensure that said data is more accurate

and thus provide a better representation of people being monitored to improve effectiveness and efficiency of preventative measures against the virus. Not only will this system design help improve accuracy and efficiency, it will also help to reduce spread of the virus by eliminating the need for a user operator for the equipment, as it is all self sufficient and automated. There would be no risk of putting the operator in danger of catching the virus as well as no possible threat of the operator spreading the virus to everyone being monitored.

2.3 Design constraints and feasibility

The main technical design constraints that had to be considered throughout the development of the system was primarily hardware related. Obviously the system design is limited by the capability of the hardware being used and as such, must be factored when designing the software for the hardware to run while in use. As previously mentioned, the hardware being considered for the system design includes the following:

1. The Mbed nxp LP1768 microcontroller
2. The TMP006 IR Temperature sensor module
3. The RN-41/Rn-42

The standard mbed compiler used for the nxp microcontroller as well as the other attached modules will be using C/C++ by default. Therefore, all of the backend coding involving specific mbed hardware functionality will be handled with C/C++.

Most if not all physical and economical constraints were due almost entirely to challenges presented by Covid. Due to the nature of the pandemic and residential conditions of group members, the project was handled remotely with the exception of one or two in person meetings with the course instructor. This drastically affected the progression and deployment of the system as the system included not only software implementation, but also hardware implementation as well as proper setup and configuration of the hardware. This in conjunction with each group given only one application board due to funding restrictions of the University made group collaboration exponentially more difficult to accomplish. Finally, each member of the group was expected to use personal finances to order the hardware equipment required for the project. Due to limitations and hardships presented by the pandemic, only one working prototype was effectively produced from collaborative efforts.

2.4 Literature and technical survey

- *FLIR Thermal Cameras*
    - Very expensive to deploy. Can cost well over $7,000 for the camera alone and an additional $4,000 for black-body radiator for a minimum of $11,000. This doesn't include the cost of associated hardware and potential additional hardware that might be needed.

- FDA has stated that temperature readings are more accurate when the target is standing still versus in motion, making thermal cameras less accurate.
- Thermal cameras tend to record in much lower resolutions than traditional cameras and thus are often unreliable when detecting fevers in moving people.

- *Berrcom Non Contact Infrared Forehead Thermometer JXB-178*
  - Typically measure a lower range of temperatures than other infrared thermal guns (i.e. ones used for home improvement).
  - Because of the way they are designed, they require measurements to be taken at less than two inches away from the desired target.
  - FDA-certified
  - Memory can store up to 32 Temperature Readings at a time.
  - Has the ability to take body temperature, ambient air temperature, and surface temperature.
  - Price ranges from $15 - $30.

- *Temperature Scanning/Screening Kiosk*
  - Expensive to deploy. Can cost upwards of $4,000.
  - Uses biometrics to map a user's facial features from a photograph or video and store them in a user profile for data to be logged.
  - Biometrics : biological measurements such as eyes or face shape that can be used to uniquely identify an individual.
  - Similar to infrared scanning tech in thermal cameras, but only scans the forehead instead of the entire body, essentially a temporal scanner.
  - Takes readings within a few seconds, making it a suitable option for mass temperature vetting purposes.

- *CIR-100 Contactless Infrared Temperature Sensor*
  - Measures core body temperature in range of 34˚ - 43˚C.
  - Has an accuracy of ±0.5˚C over a full measurement range of 25-75cm when used indoors.
  - Fast temperature measurements completed within 2 seconds.
  - Scans face and uses 2m's smart algorithms to compute oral body temperature.

- Combines infrared sensing techniques with laser based photonics.
- Connected via a USB interface to a computer, which handles power and communication.
- Price unavailable.

- *Melexis MLX90632*
  - Non-contact FIR sensor that is significantly smaller in size than competitive hardware at just 3x3x1 mm surface mount.
  - Highly accurate at rate of ±0.2˚C straight out of factory calibration.
  - Uses a suspended membrane only a few micrometers thick that respond very quickly to surface heat radiation.
    - Designed to have very low thermal mass and is thermally isolated from the rest of the chip.
  - Uses I2C connection with the host system.
  - Small size makes this device extremely portable and ideal for wearable and hearable devices.
  - High portability makes continuous body temperature monitoring highly probable.
  - Very affordable at around $15-$25 dollars per single chip.
  - Requires additional hardware to use properly.

- Conclusion:

It is apparent that several of the options that work right out of the box are far more expensive than the proposed mbed system design, with the kiosk coming in over $4,000 and the kiosk totalling over $11,000. Some of the other cheaper solutions do seem more attractive at first glance. However, one must consider that for the Berrcom thermal gun, a user is required to operate the hardware which, as mentioned earlier, poses a health risk to both the user and the people being tested/monitored. Also, other cheaper solutions like the Melexis MLX90632 requires additional hardware to use properly and is not a stand-alone unit. Thus we can conclude that the mbed system design is still the overall superior system design. If there is indeed a way to effectively use the mbed chip in conjunction with the Melexis sensor, this might prove to be the optimal system design for cost/power efficiency.

- Sources:
    - https://blinkidentity.com/contactless-fever-scanning-what-you-need-to-know/
    - https://www.bhphotovideo.com/c/product/1560921-REG/berrcom_jxb_178_non_contact_infrared_thermometer.html
    - https://www.soscanhelp.com/blog/what-is-a-temperature-scanning-kiosk
    - https://www.2mel.nl/projects/medical-infrared-temperature-sensor/
    - https://www.techbriefs.com/component/content/article/tb/supplements/st/features/technology-leaders/37011

2.5 Evaluation of alternative solutions

The project design given to us was straightforward and concise as to exactly what the specifications and requirements were. As such, there were no other alternatives presented to the group or considered by anyone involved in the project. Had there been other options or alternatives in terms of choices that could have been made differently, the main variation that could have taken place would be the use of slightly different hardware. This would primarily entail switching out specific hardware components such as the microcontroller that was used. There are highly capable alternatives that could have been used such as an Arduino microcontroller as well as the current/earlier generations of the Raspberry Pi single board computer.

3 Final design:

3.1 System description

The overall system has 5 main components. These components include the following:

1. Mbed NXP LPC1768 microcontroller.
2. RN41/RN42 bluetooth module chip.
3. TMP006 IR Temperature sensor.
4. Mbed Application board or equivalent hardware (what is used to connect all hardware components together).
5. Host computer and/or mobile host (i.e. phone/tablet/laptop).

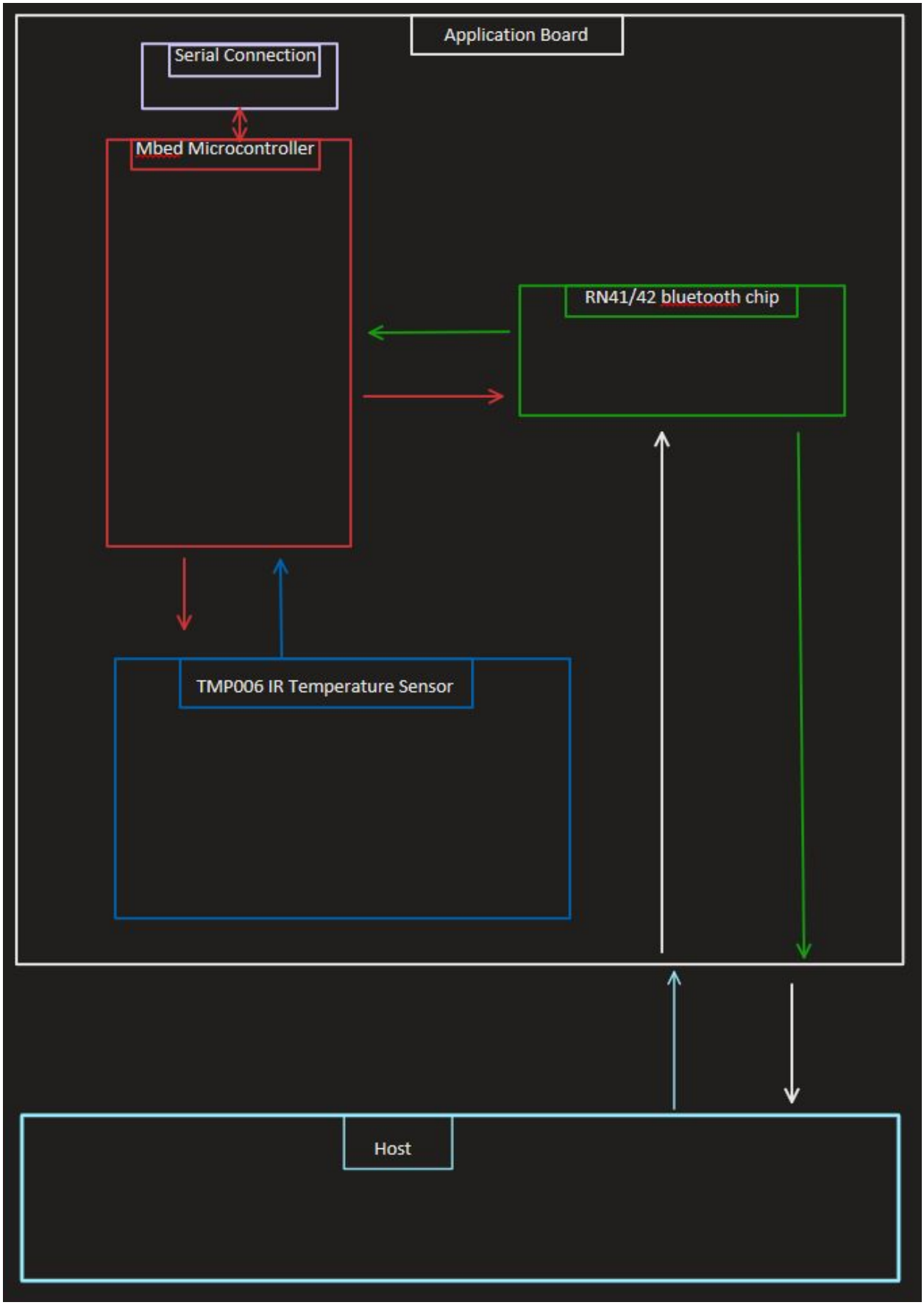
The system is designed to work in the following fashion:

First and foremost, the Mbed NXP LPC1768 microcontroller as well as the TMP006 IR Temperature sensor are both plugged into the application board with their corresponding pins respectively. Next, the RN41/RN42 bluetooth chip module must have 4 wires soldered onto its ground, power, Tx, and Rx pins. These wires will each have a male connector at the wires end which will plug into a double female wire that is connected to the ground and Vout pins as well as pin 9 and pin 10 on the Mbed microcontroller. After all connections are successfully made and the microcontroller as well as the other hardware components/modules are successfully powered, the system is ready to receive the software it will use for deployment.

The Mbed microcontroller contains a mini-USB port that acts as both a power supply as well as a serial port connection. Appropriate software is compiled and then uploaded via USB serial port connection using the provided mbed compiler. Once the software has been uploaded and compiled successfully, the user will need to press the one blue button located towards the middle-bottom of the Mbed microcontroller in order to compile the software on the board. The board will store all software uploaded to it on the onboard flash memory. Assuming that the temperature sensor has been properly connected, software may be uploaded to test this as well and will be configurable via the mbed compiler.

The next step that must take place before system deployment is proper configuration of the bluetooth chip module. This can and must be properly configured using the Tera Term terminal emulator or equivalent software such as “Putty” so that the chip can communicate properly with the Host device. Detailed steps to properly configure this module are provided in the RN41/RN42 documentation provided by Roving Networks. Once proper configurations have taken place and the bluetooth chip is able to print and send data to the host device via bluetooth connection alone, the system is theoretically ready for deployment if all components and software are behaving as they should.

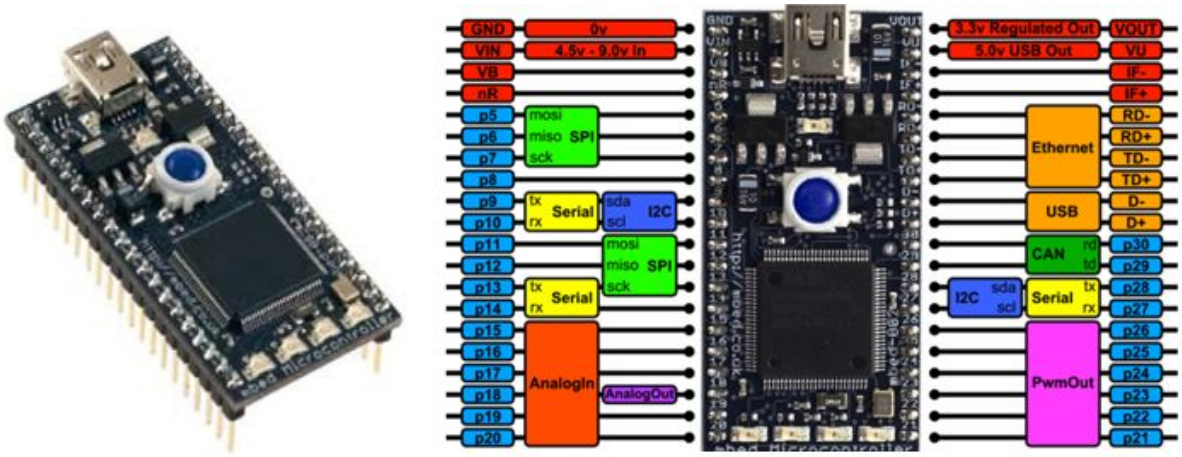
The design located on the following page is a high level block diagram that accurately displays the layout and functionality of all components comprising the complete system design as well as how they effectively communicate with each other while in use :

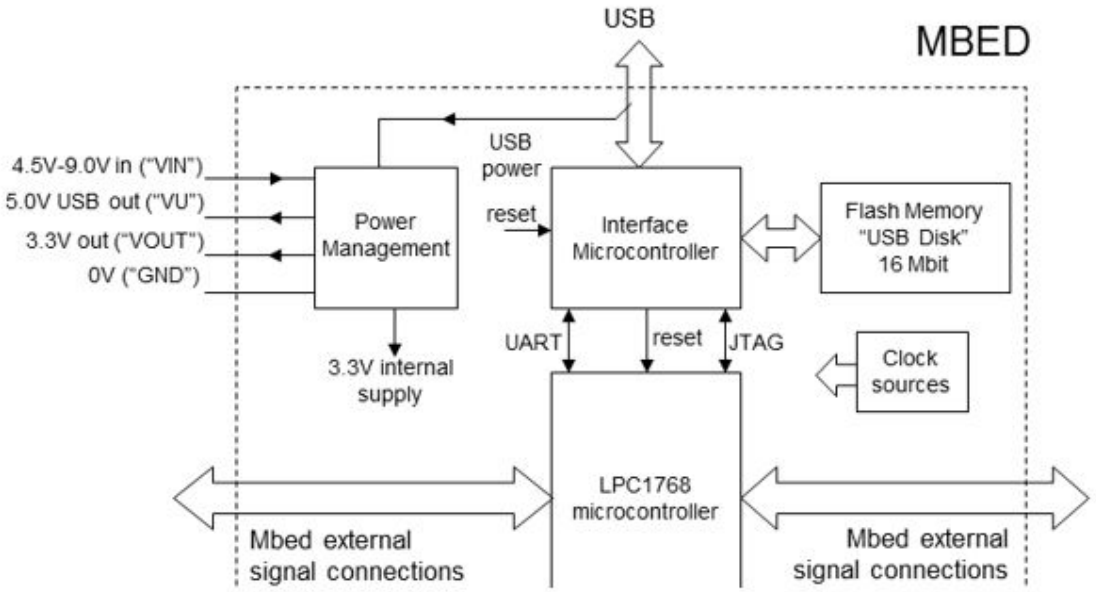

Application Board
Serial Connection
Mbed Microcontroller
RN41/42 bluetooth chip
TMP006 IR Temperature Sensor
Host

3.2 Complete module-wise specifications

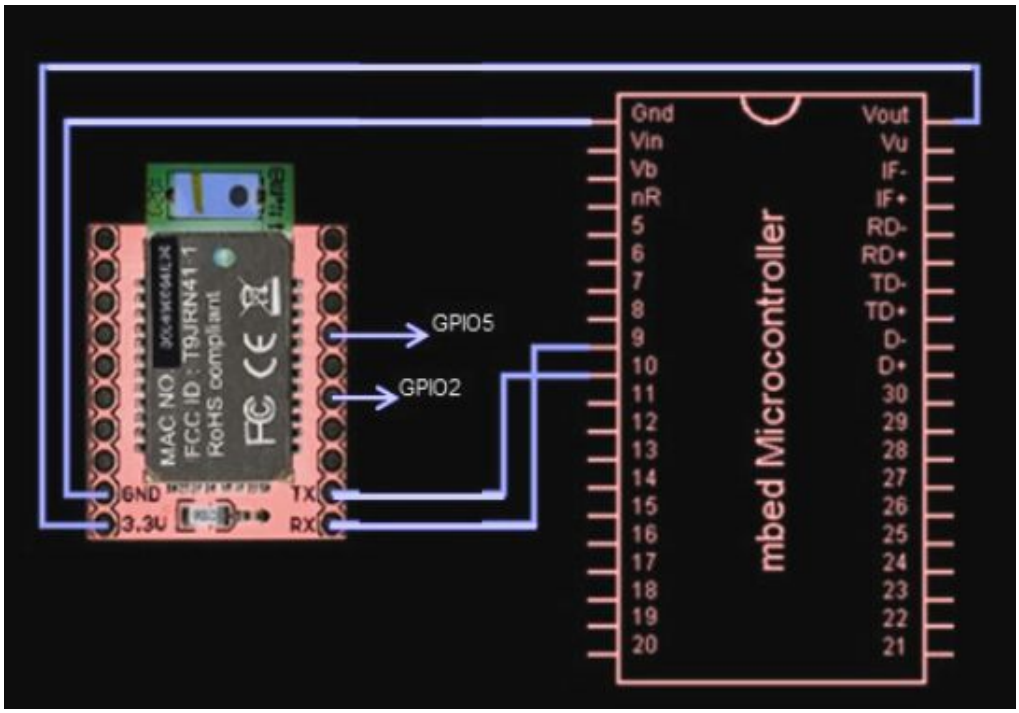
Circuit and logic diagrams, interfaces and pin-outs:

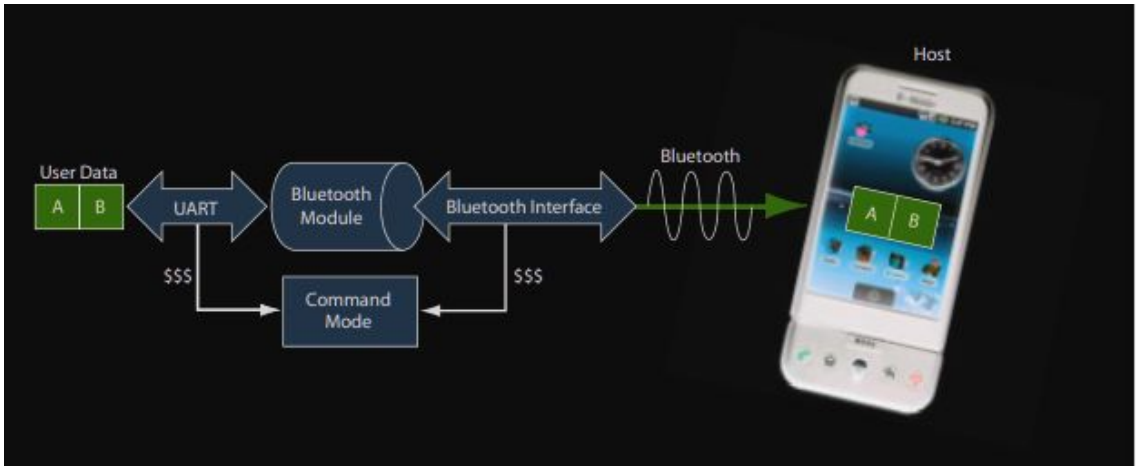
The Mbed Nxp LCP1768 microcontroller:



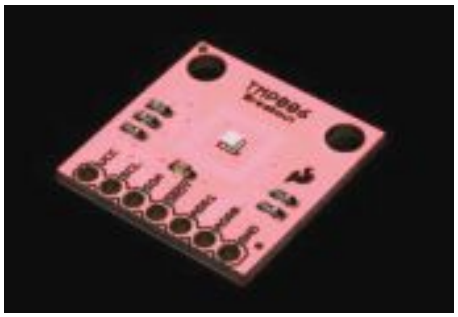


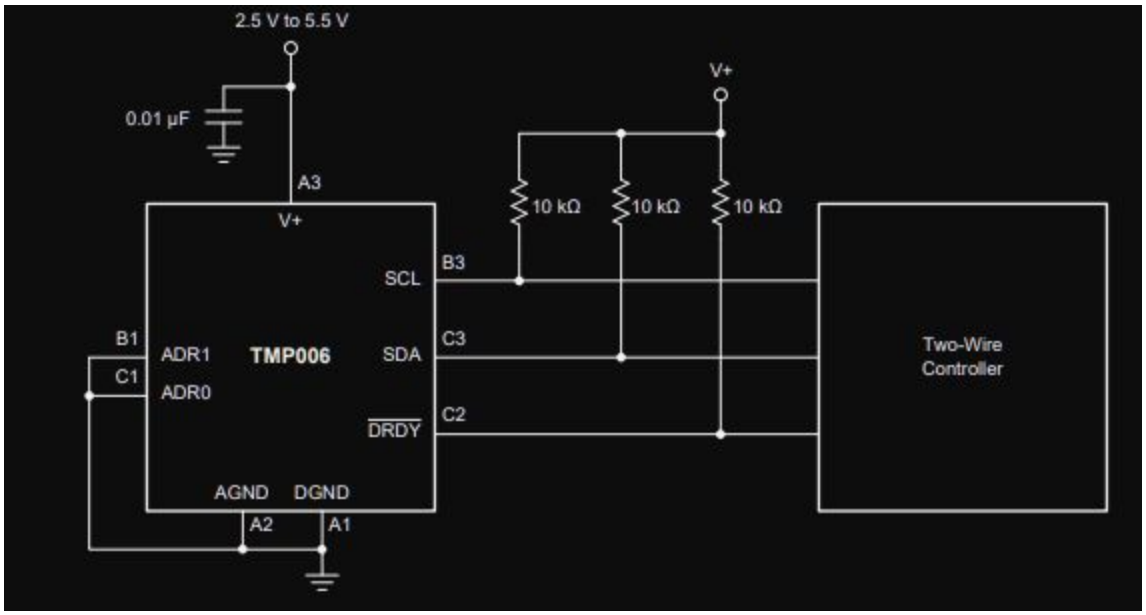
The RN41/RN42 bluetooth module chip:

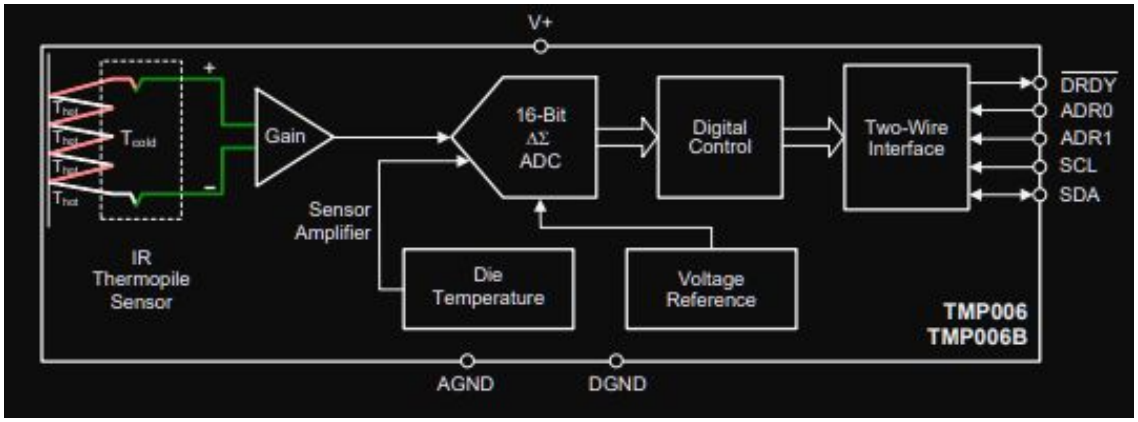




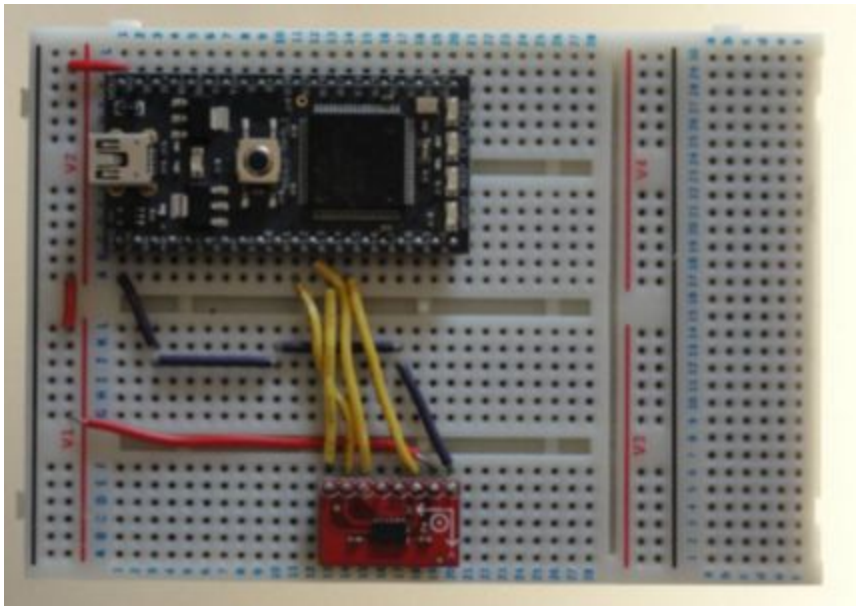
The TMP006 IR Temperature sensor:

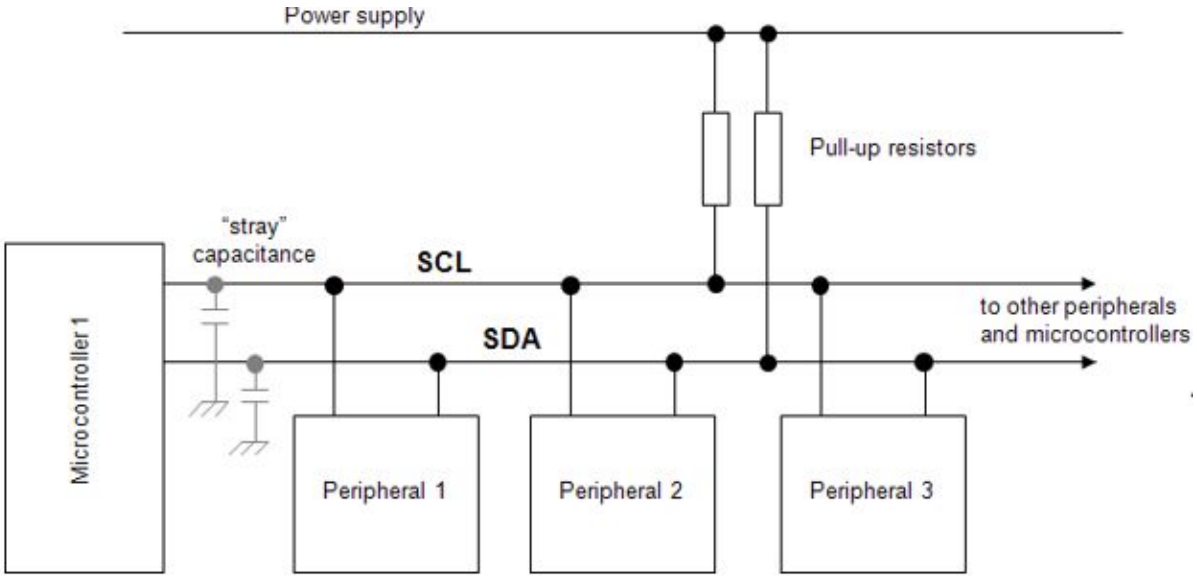






The Application Board:





- Complete parts list:
    - Mbed NXP LPC1768 microcontroller.
    - RN41/RN42 bluetooth module chip.
    - TMP006 IR Temperature sensor.
    - Mbed Application board or equivalent hardware (what is used to connect all hardware components together).
    - Host computer and/or mobile host (i.e. phone/tablet/laptop).

3.3 Approach for design validation

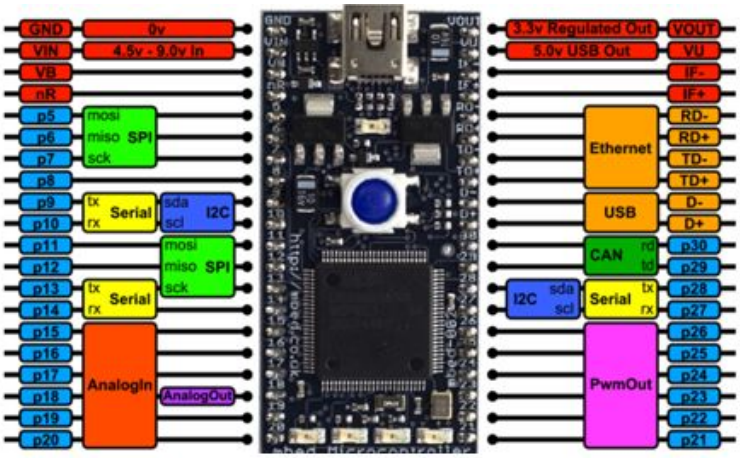
The system was tested for effectiveness and reliability using the following steps:

First, the board and components were hooked up via serial connection to effectively provide a power supply to all the hardware components including the Mbed microcontroller as well as the temperature sensor and bluetooth module. Doing this also confirmed that the Application board was properly working as well as the soldered connections of the bluetooth chip. Next, code from either specific component API's or example code was uploaded to the Mbed and tested to make sure hardware components were functioning as expected. Once this was completed all hardware was successfully tested by gaining readings and measurements from specified components. This process was then repeated after uploading original code that needed to be uploaded in conjunction with certain API calls that were used.

4 Implementation notes

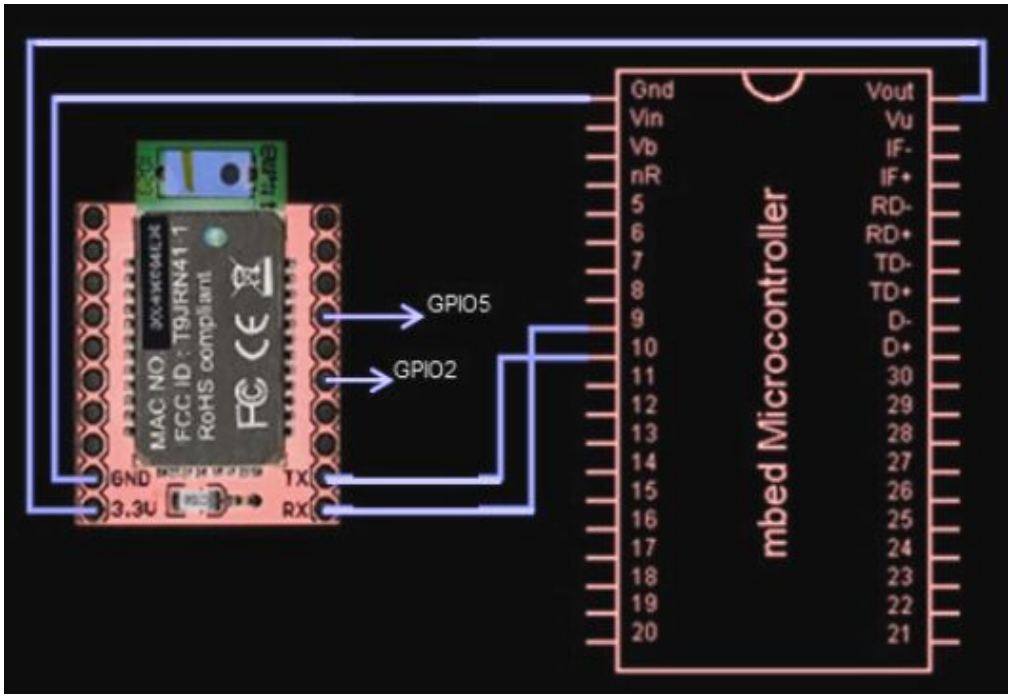
As previously mentioned, the overall system design and complexity is relatively low, containing just 4 main components if the host pc/mobile device/laptop is excluded as one would assume someone would at minimum have a smartphone capable of hosting. These components, as stated before, include the Mbed NXP LPC1768, the RN41/42 bluetooth chip module(either model will suffice), the TMP006 IR Temperature Sensor, and Mbed Application or equally capable board (Mbed's house Application board is no longer available for purchase from their website).

The Mbed microcontroller acts as the brains and interface of the system by allowing user uploaded software to enable hardware components to function properly as well as the specific pins of the controller providing specific functionality for hardware requirements and user preferability.



As seen previously, this diagram shows the various pins of the Mbed microcontroller and what they are used for. For this system, we will using the GND and Vout pins to establish a circuit. Next we will be using p27 and p28 respectively for the RN41 bluetooth chip connection. Finally, the TMP006 temperature sensor is hooked up to p9 and p10 respectively. Interchangeability between serial communication pins for specific hardware components is acceptable as there are 3 pairs of viable pins labeled in the diagram above.

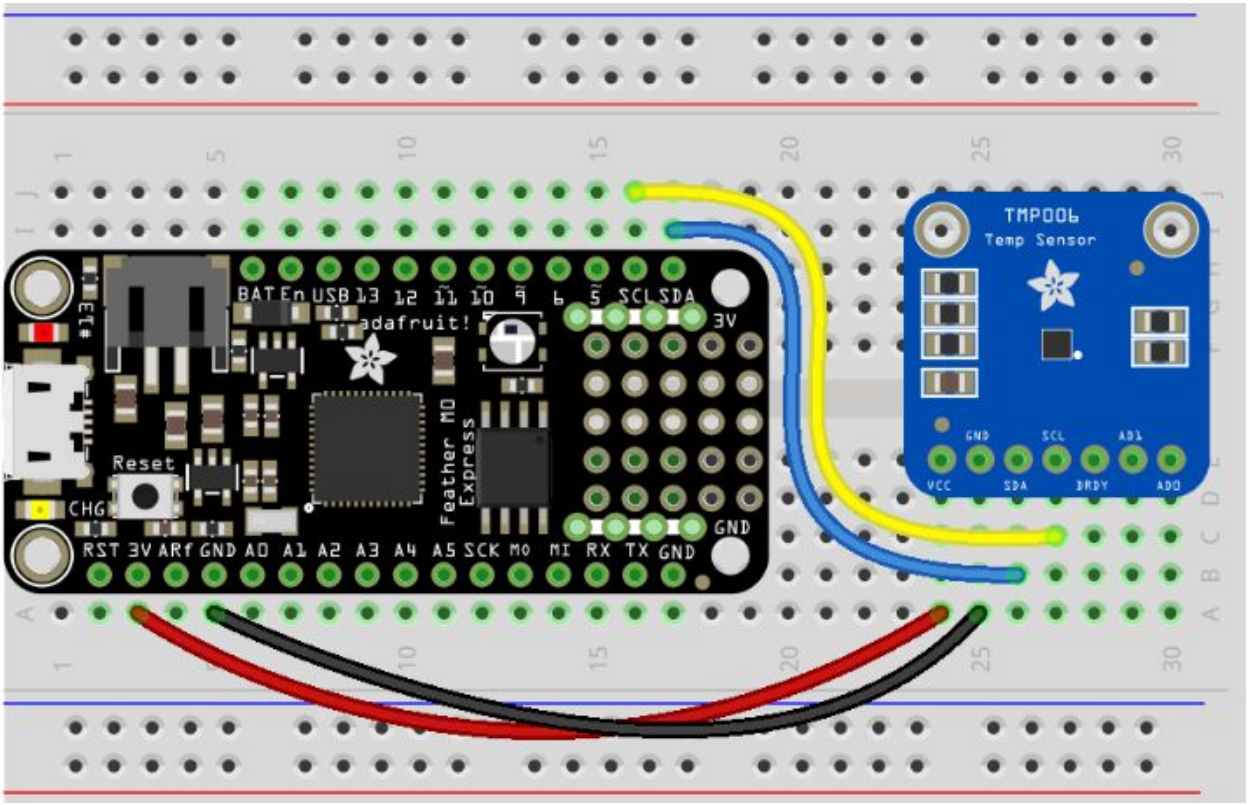
The RN41 bluetooth chip acts as a bridge of communication between the host and the board as well as capability to connect to other boards to establish a local network of communication. Furthermore, the bluetooth chip will connect to a host device that presumably has internet access/connection effectively making the system an IoT device.



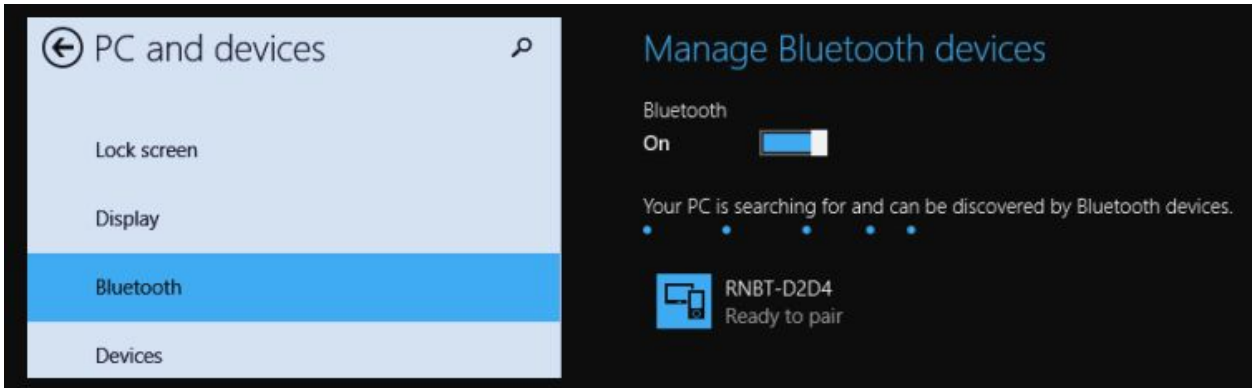
From observing the above diagram, we can see exactly what pins from the bluetooth chip module correspond to the various pins on the mbed microcontroller. On the left side of the bluetooth chip we can see the very bottom one is the 3.3V power supply pin that needs to be connected to the Vout on the top right of the microcontroller. Similarly we can see that the pin above the 3.3V pin labeled GND for ground needs to be connected to the corresponding GND pin located on the top left of the mbed microcontroller. On the bottom right of the bluetooth chip we can see the TX and RX pins which can be hooked up to any of the three pairs of aforementioned serial communication pins on the microcontroller. In this diagram, pins 9 and 10 are selected to hook up these TX and RX pins. It should be noted that for making the connection to the pins on the bluetooth chip, wire will need to be successfully soldered to make contact with these pins, however for the connection with the microcontroller, normal connections to the pins without soldering is perfectly fine.

The TMP006 IR Temperature Sensor uses infrared technology to scan the intended target for temperature within a 0.5 degree C accuracy. This sensor is contactless and uses a very sensitive thermopile to measure the infrared energy being emitted by the targeted object. This sensor works with a 3v to 5v and connects via the i2c bus and as such, is addressable. Therefore, it is possible to have up to 8 TMP006 sensors on the same bus. A thermopile, as briefly mentioned, is a multitude of thermocouples aligned in parallel, but wired in series. Each

thermocouple will generate a microvolt- level signal proportional to the temperature differential from the hot end to the cold end. By wiring them in parallel, the output is the sum of all outputs of all the thermocouples. Very small temperature differences can be measured properly and effectively by multiplying the output in this fashion. Unfortunately, a proper wiring diagram using the Mbed LPC1768 microcontroller is not available, however a very similar diagram using the Feather MO is with I2C on an application board which will be shown below:

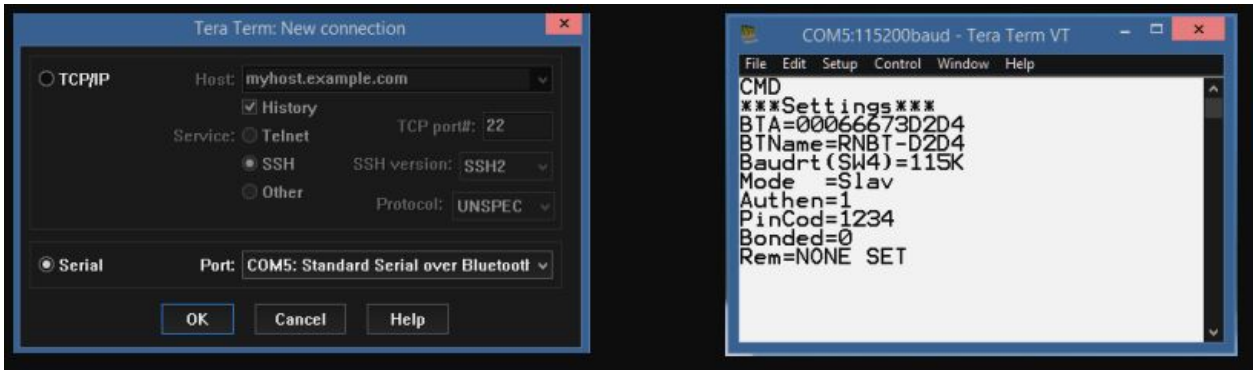


Once the hardware components have been successfully connected to the microcontroller on the application board, certain components will be plug and play and work right after a connection is made and accommodating software api is uploaded to the microcontroller. However, components like the bluetooth chip module are a little different. Some initial set up will be required before being able to test to make sure connection has been properly made with the solder and microcontroller. The following images shows specific steps that will ne to be taken before testing the bluetooth chip:



As we can see by the image, we will first need to pair the bluetooth chip with the pc/mobile device that will be set up to be the host for the system design. Once successful pairing has taken place, Tera Term, an open-source free software implemented terminal emulator will need to be downloaded at the following link : https://osdn.net/projects/ttssh2/releases/

After downloading the appropriate version of Tera Term, it will need to be launched and have the Serial option selected with the corresponding com port selected that matches the outgoing bluetooth connection to the RN41 chip. You can see an example of this here:



On the right is what should be seen if entering the command mode of the bluetooth chip. This can be done by pressing the '$' character three times like so "$$$".

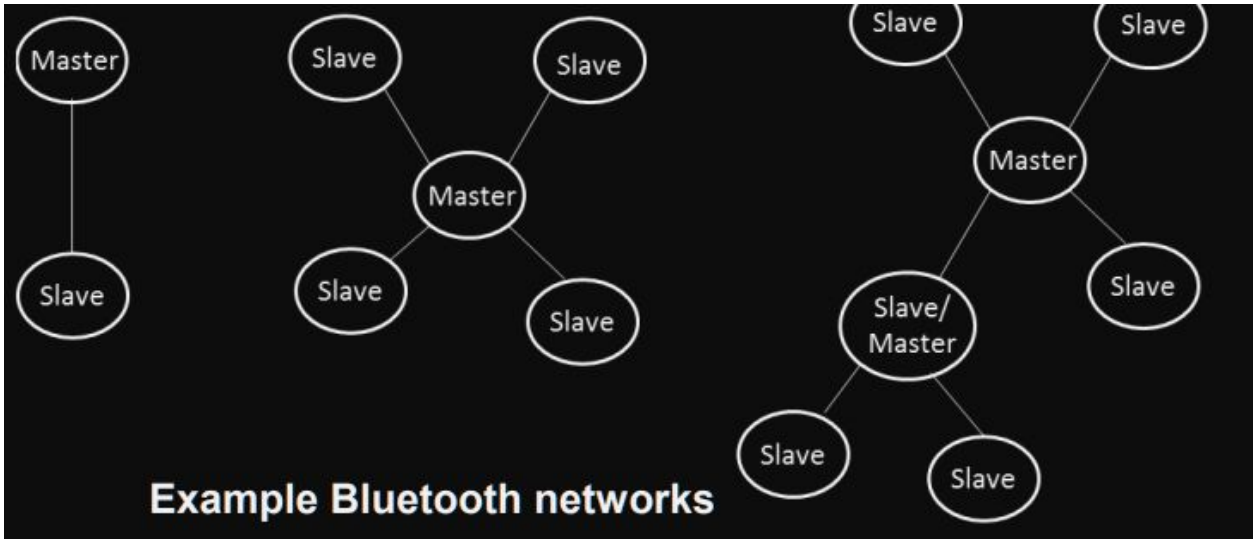
It will work with example code after making the proper connection, but it will need to be properly configured. For instance, after uploading the following example code, the bluetooth chip will behave as expected, confirming that the bluetooth has been initially setup correctly:

```
/* Program Example 11.2: Bluetooth serial test data program
   Data is transferred bidirectionally between mbed and PC via
Bluetooth.
                                                              */

#include "mbed.h"
Serial rn41(p9,p10);                //name the serial port rn41
BusOut led(LED4,LED3,LED2,LED1);

int main() {
  rn41.baud(115200);            // setup baud rate
  rn41.printf("Serial data test: outputs received data to
LEDs\n\r");
  while (1) {
    if (rn41.readable()) {      // if data available
      char x=rn41.getc();       // get data
      led=x;                    // output LSByte to LEDs
    }
  }
}
```

As the description says, this sample program tests bidirectional data flow between the mbed microcontroller and host pc. What it should do, if working properly, is light up the LEDs on the microcontroller in correspondence to keyboard character presses data received by the RN41 chip when the user types anything into the outgoing com port of the Tera term terminal emulator. Furthermore, the microcontroller can be either plugged into another computer or straight into a power adapter, effectively only providing power and not a serial connection to the bluetooth chip. This is how effectiveness and proper set up has been achieved up the bluetooth module.

Unfortunately, in order to print data on the host pc via the bluetooth chip from the microcontroller, further settings will need to be configured. If we look at the full documentation of the chip provided by Roving Networks, we can see the exact steps and configuration settings that need to be made. Caution is highly recommended when making any configuration changes, as certain changes made can render the bluetooth chip into a mode that will not be usable/ideal for the system design we are trying to use. This is especially true when determining which mode

the chip will be using. For instance, if multiple mbed microcontrollers are being used within the system, different modes will be used by certain controllers in order to establish a properly configured master slave network of sorts. An example diagram is shown here:



Furthermore, we can see on the following figure on the next page that we have a variety of different modes that we can set the chip into with different effects and behaviors of the chip. It should be duly noted that by setting the chip into certain modes, you will not be able to enter the command mode which will dramatically affect the way your chip behaves. If this is done and there is only one complete system, the chip will potentially not be able to re enter command mode to switch it into a different mode that will allow for entering Command mode. Roving Networks provides this warning at the top of the figure as well.

1.4 OPERATING MODES

The Bluetooth module has several operating modes, which you set using the SM command in command mode.

> **Note:** In all master modes, the module cannot be discovered or configured remotely over Bluetooth.

- *Slave Mode (SM,0)*—Default mode, in which other Bluetooth devices can discover and connect to the module. You can also make outbound connections in this mode.
- *Master Mode (SM,1)*—In this low-speed connection mode, the module makes connections when a connect command (C) is received. This command can also contain the Bluetooth address of the remote device. If a device is not specified, the module uses the stored remote address. The connection can be broken if the special break character or string is sent (use the SO command to set the break character). This mode is useful when you want the module to initiate connections (not receive them). In this mode, the module is NOT discoverable or connectable.
- *Trigger Mode (SM,2)*—In this low-speed connection mode, the module makes connections automatically when a character is received on the serial port (UART). The connection continues as long as characters are received on either end. The module has a configurable timeout (which you set using the ST command) that disconnects the module after the specified number of seconds of inactivity (1 to 255) or a configurable break character is received.
- *Auto-Connect Master Mode (SM,3)*—In this mode, the module makes connections automatically on power-up and re-connects when the connection is lost. This mode can be set by command, or by setting the external dipswitch 3 during power up (evaluation kits) or by driving GPIO6 high (Bluetooth modules). If an address is not stored, the module performs an inquiry process and the first device found that matches the COD is stored. In this mode, high-speed data is passed without being interpreted; therefore, the connection cannot be broken via commands or software break characters. If a disconnect occurs, the module attempts to re-connect until successful.
- *Auto-Connect DTR Mode (SM,4)*—This mode must be set by command. It operates like Auto-Connect Master Mode, except that you control connection and disconnection with dipswitch 3 (evaluation kits) and GPIO6 (Bluetooth modules). Turning the dipswitch on or driving GPIO6 high initiates the auto-connect process; turning the dipswitch off or driving GPIO6 low causes a disconnect.
- *Auto-Connect ANY Mode (SM,5)*—This mode must be set by command. This mode operates like Auto-Connect DTR Mode, except that each time the dipswitch or GPIO is set, an inquiry is performed and the first device found is connected. The stored address is NOT used, and the address found is never stored.
- *Pairing Mode (SM,6)*—In this mode, the module attempts to connect with the remote device matching the store remote address. You set the remote address using the SR command.

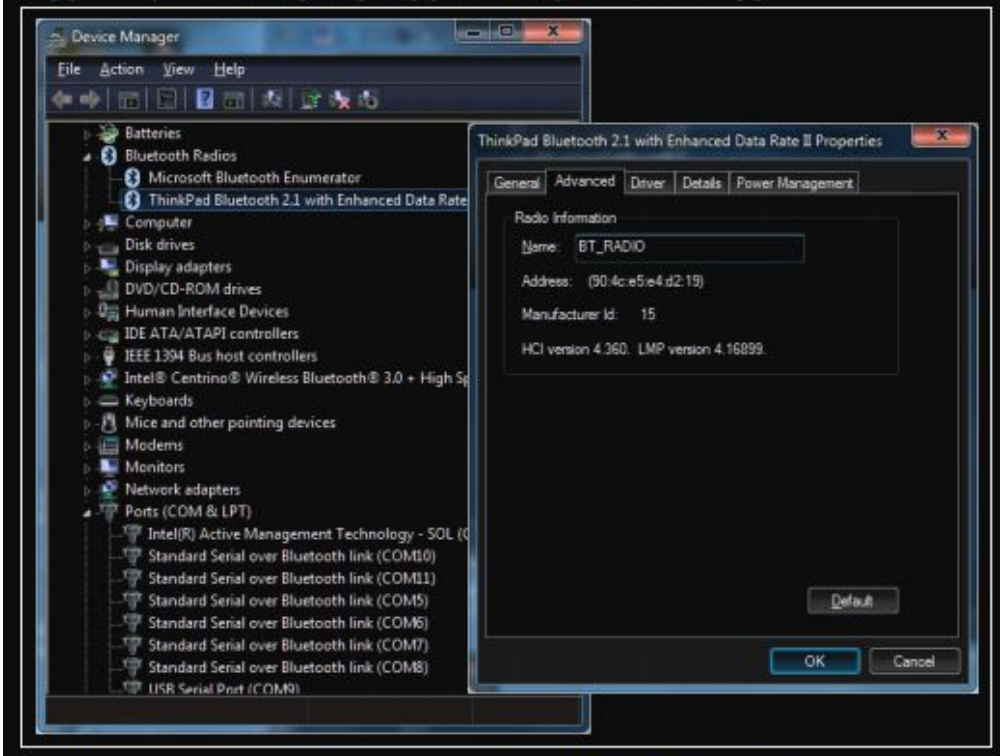
If the bluetooth chip happens to be configured in one of the above modes that disables the ability to enter command mode and there is no other devices that are in the system network, there is actually a way to factory reset the chip as seen by the documentation in the figure below:

Function	Dipswitch (Adpaters & Evaluation Boards)	GPIO Pin (Modules)	Settings (OFF = 0 VDC/ON = 3 VDC)
Factory Reset	1	GPIO4	Off = disabled, on = armed. Set this dipswitch/GPIO pin on power up to arm the reset function. Then toggle the module off and on three times to reset all settings to the factory defaults (other than the Bluetooth name).
Auto Discovery/ Pairing	2	GPIO3	Off = disabled, on = enabled. You use these settings in conjunction with dipswitch 3/GPIO6. If dipswitch 3/GPIO6 are also set, the module performs a device inquiry scan, searching for a partner device with a special matching class (0x55AA). Once it finds this device, it stores the address into the remote address field and auto-connects to the remote device. If dipswitch 3/GPIO6 are NOT set, the module enters slave mode with the special matching class and waits for the master to find it. This mode is usually set once on both ends of a module pair (for instant cable replacement) and then removed.
Auto-Connect	3	GPIO6	Off = disabled, on = enabled. This setting is equivalent to Auto-Connect Master Mode in software. The module connects to the stored address. If dipswitch 2/GPIO3 is also set, a new discovery/pairing can be made. If connected via the CFR command, toggling the dipswitch off-on-off terminates the current connection.
Baud Rate	4	GPIO7	Off = stored setting (115 K), on = 9,600. This setting is used to configure 9,600 or a software selected (default = 115 K) baud rate. If the dipswitch is off, the module uses the stored baud rate setting. When the dipswitch is on, the baud rate is set to 9,600 regardless of the software setting.

If a wire is soldered to the GP104 pin on the bluetooth chip and has a dipswitch connected as well as having power supplied to it, the switch can be toggled three times in succession in order to factory reset the chip back to its original default configuration settings. Similarly, you can do this same process to varying other pins as seen above to hard configure certain settings of the chip such as discovery and pairing mode, auto-connection, and baud rate settings.

Once everything has been connected and configured correctly, you should be able to do the following steps and show by the figures to allow for successful communication and printing over bluetooth connection from the microcontroller system to the host pc.

FIGURE 1-6: PC'S BLUETOOTH RADIO MAC ADDRESS

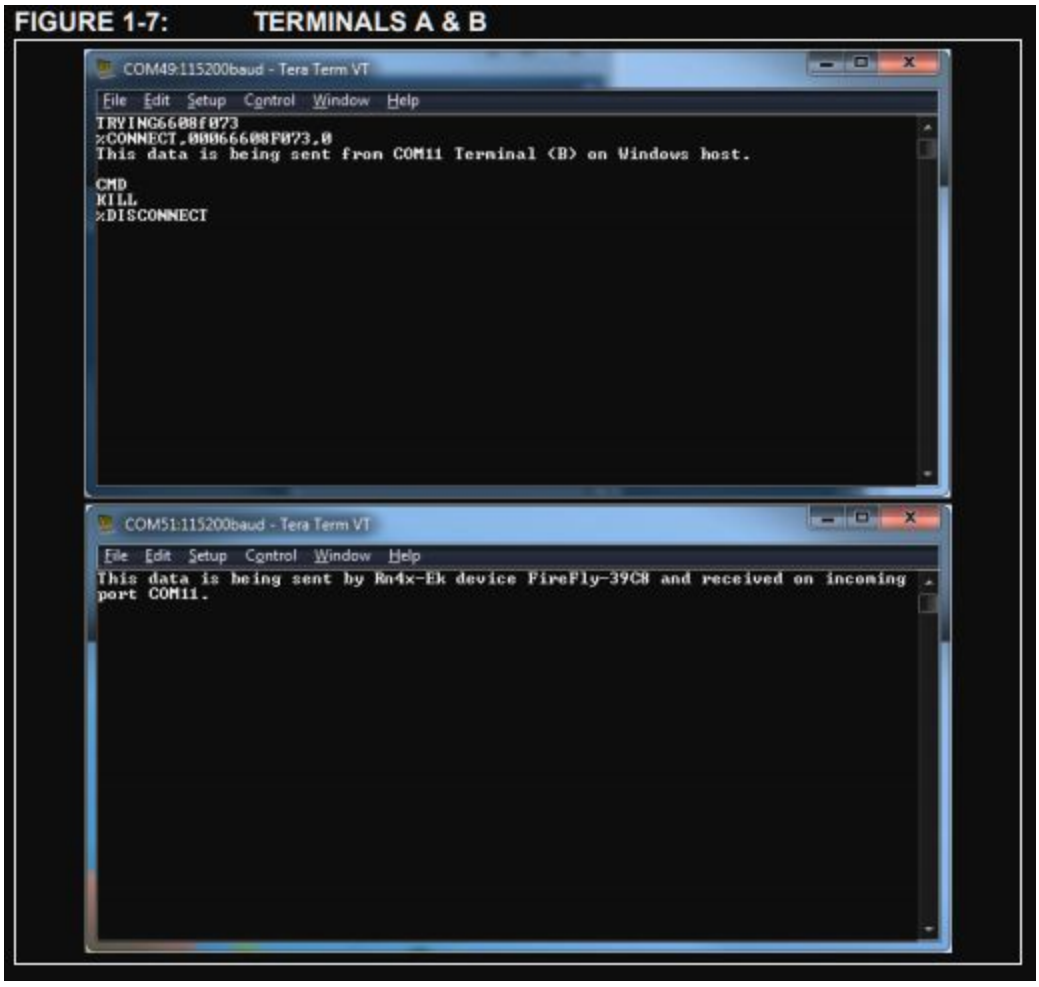


2. Pair your module with the PC as described in "Pairing" on page 13.
3. Open a terminal (called terminal A in this example) and connect it to the module. You can run this terminal on the host PC or another computer.
4. Open a second terminal (called terminal B in this example) on the host PC to listen for the incoming Bluetooth connection using the incoming COM port number.
5. Type `C,`*<MAC address>* <cr> in terminal A to establish an SPP connection to the host PC. See Figure 1-7 for an example connection.
6. Try the following commands:
   - `$$$` to enter command mode
   - `SO,%` to enable status message to see connect/disconnect conditions
   - `R,1` to reboot
   - `$$$` to re-enter command mode
   - `+` to enable local echo
   - `C,`*<MAC address>* to attempt a connection with remote device

   Characters you type in terminal B are sent over Bluetooth to the host PC and appear in terminal A. Any characters entered in terminal A are transmitted to terminal B.
7. To kill the connection, type the `k,1` <cr> command in terminal B.

After successfully completing the steps in the above figure, your output should look something like this:

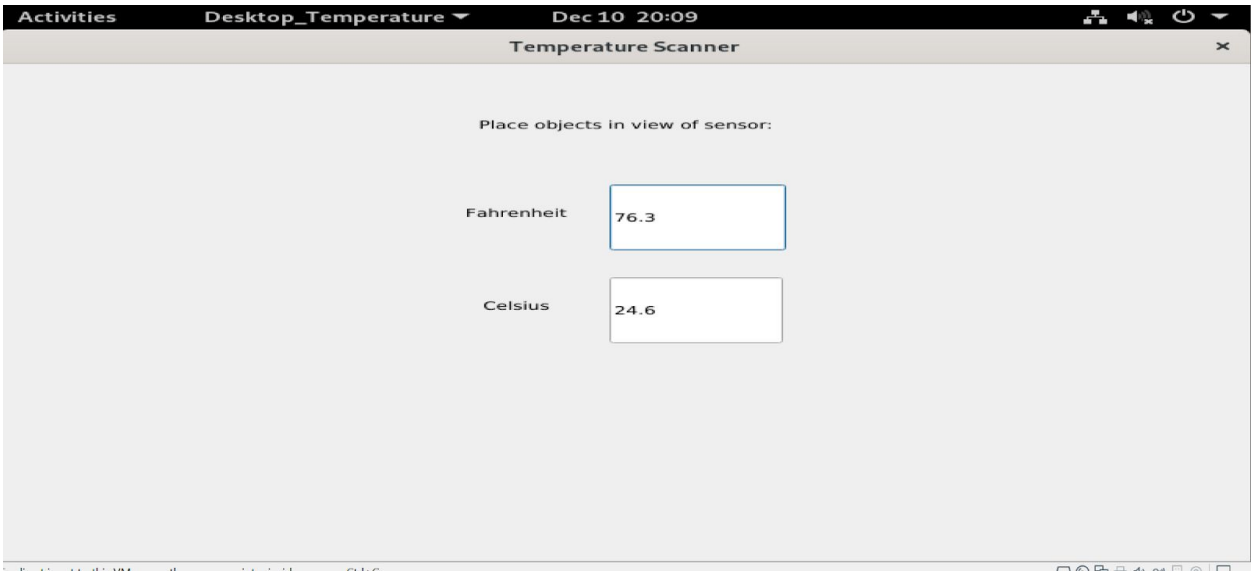
FIGURE 1-7: TERMINALS A & B

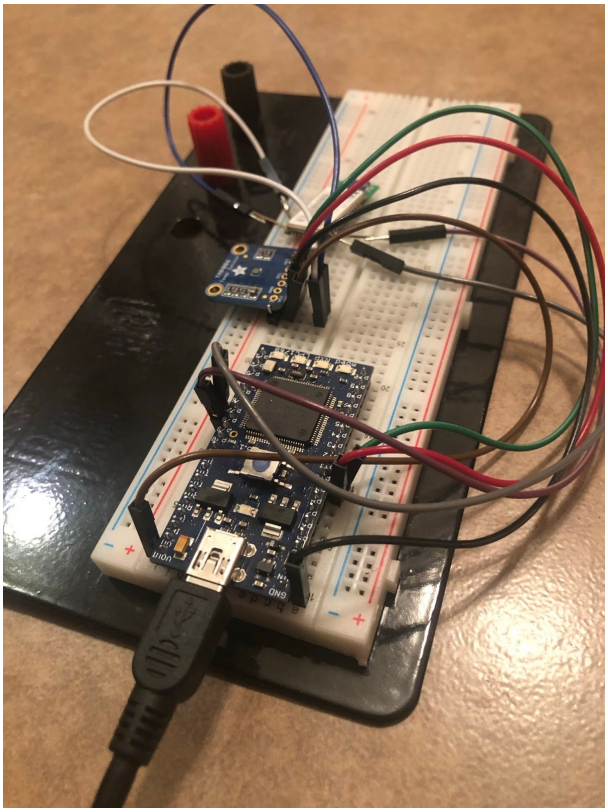


Note that these steps will vary slightly for Mac users as well as if trying to set up the system on a mobile host device.

The security used for the temperature's taken and the user's information was from the Mbed Studio's own Security API, Mbed TLS via authenticated encryption. The encryption receives plaintext and a key and encrypts the plaintext into a ciphertext and an authentication tag. The decryption receives the ciphertext, the key, and the authentication tag and decrypts it back into a plaintext. If the authentication tag does not match the ciphertext, then an error will take place of the plaintext.

5 Experimental results





main.cpp > main(void)

```
#include "no_contact_temp.h"
#include "TMP006.h"

#include <stdio.h>
#include <string.h>

using namespace std;

//for TMP006
#define Address             0x80
#define TMP006_TX_PIN       p9
#define TMP006_RX_PIN       p10

//BufferedSerial
#define RN41_TX_PIN         p28
#define RN41_RX_PIN         p27
#define MAXIMUM_BUFFER_SIZE 30

DigitalOut led1(LED1);
DigitalOut led2(LED2);
DigitalOut led3(LED3);
DigitalOut led4(LED4);

TMP006 sensor(TMP006_TX_PIN, TMP006_RX_PIN, Address);

static BufferedSerial rn41(RN41_TX_PIN, RN41_RX_PIN, 115200);
static BufferedSerial pc(USBTX, USBRX, 9600);
/*default is 9600 baud, 8 bit data data fram, no parity, one stop bit(1)*/

int main(void)
```

Temp_Sensor_Desktop > temperature.cpp > Temperature::getTemp()

```
96      bool start_reading = false;
97      //start at index 5 to avoid a number that was split
98      for(int i = 0; i<READ_BUFFER_LENGTH; i++){
99          if (read_buf[i] == '$'){ // recognize separation char
100             if(start_reading){
101             printf("Breaking loop at %i...\ncurrent char: %c\nprevious char: %c\n", i,read_buf[i], read_buf[i - 1]);
102             if (i != READ_BUFFER_LENGTH) printf("next char : %c", read_buf[i + 1]);
103                 break;
104             }
105             else start_reading = true;
106         }
107         else{
108             if(start_reading){
109                 num_buf[buf_index] = read_buf[i];
110                 buf_index++;
111             }
112         }
113     }
114     printf("read_buf:\n%s\n", read_buf);
115     printf("num_buf:\n%s\n", num_buf);
116     close(serial_port);
117     double d;
118     try {
119         d = std::stod(num_buf);//convert string trapped in num_buf to double
120     }
121     catch{
122         d = 0.0;
123     }
124     return d;
```

Temp_Sensor_Desktop > mainwindow.cpp > MainWindow::MainWindow(QWidget *)

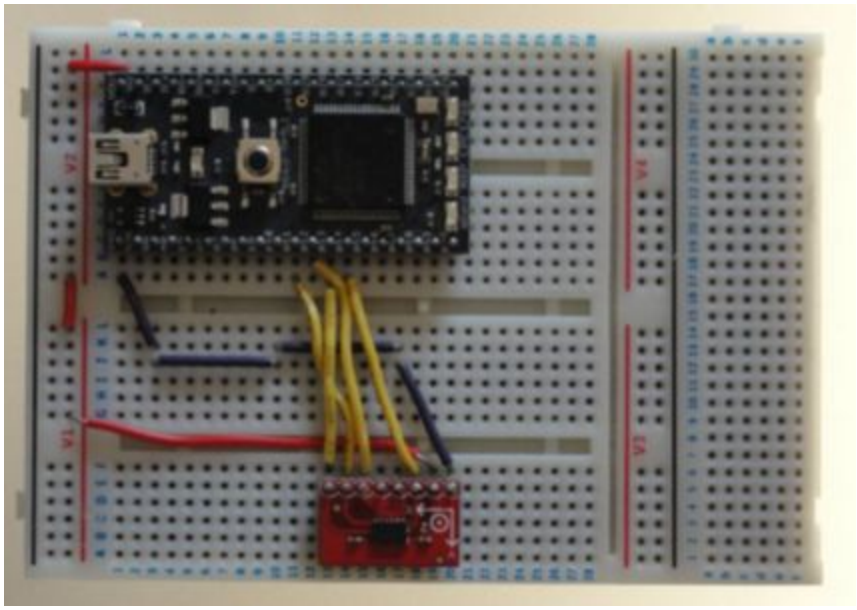
```
9   MainWindow::MainWindow(QWidget *parent)
10      : QMainWindow(parent)
11      , ui(new Ui::MainWindow)
12  {
13      ui->setupUi(this);
14      //give serial communication a head start
15      std::this_thread::sleep_for(std::chrono::milliseconds(200));
16      //timer to update temperature readings
17      QTimer *timer = new QTimer(this);
18      connect(timer, &QTimer::timeout, this, &MainWindow::UpdateTemp);
19      timer->start(200);
20  }
21
22  MainWindow::~MainWindow()
23  {
24      delete ui;
25  }
26
27  void MainWindow::UpdateTemp()
28  {
29      Temperature tr;
30      //get value as Fahrenheit
31      double f = tr.getTemp();
32      //convert to Celsius
33      double c = (f - 32.0) * 5.0 / 9.0;
34      //format output to only one decimal place
35      int c_int = int(c * 10.0);
36      c = double(c_int) / 10.0;
37      ui->f_temp_display->setText(QString::number(f));
38      ui->c_temp_display->setText(QString::number(c));
```

```
static BufferedSerial rn41(RN41_TX_PIN, RN41_RX_PIN, 115200);
static BufferedSerial pc(USBTX, USBRX, 9600);
/*default is 9600 baud, 8 bit data data fram, no parity, one stop bit(1)*/

int main(void){
    //buffer to send the temperature as a string
    char buf[MAXIMUM_BUFFER_SIZE];
    //format of string
    char format[] = {"$%i.%i$\n"};
    while (1) {
        int R; //used to print the errors
        if((R=mbedtls_platform_setup(NULL))!=0)
        {
            printf("Platfrom init failed w/ error%ds\n", R);
        }
        //set all data in buffer equal to easy to use character
        memset(&buf, '\0', sizeof(buf));
        double current_temp = sensor.readObjTempF(Address);
        int i = int(current_temp);
        int d = abs(int((current_temp - double(i)) * 10.0));
        sprintf(&buf[0], format, i, d);
        pc.write(buf, sizeof(buf));
        if(current_temp > 70.0) led1 = 1; else led1 = 0;
        if(current_temp > 80.0) led2 = 1; else led2 = 0;
        if(current_temp > 90.0) led3 = 1; else led3 = 0;
        if(current_temp >100.0) led4 = 1; else led4 = 0;
        ThisThread::sleep_for( 150ms );
    }
}
```

6 User's Manuals

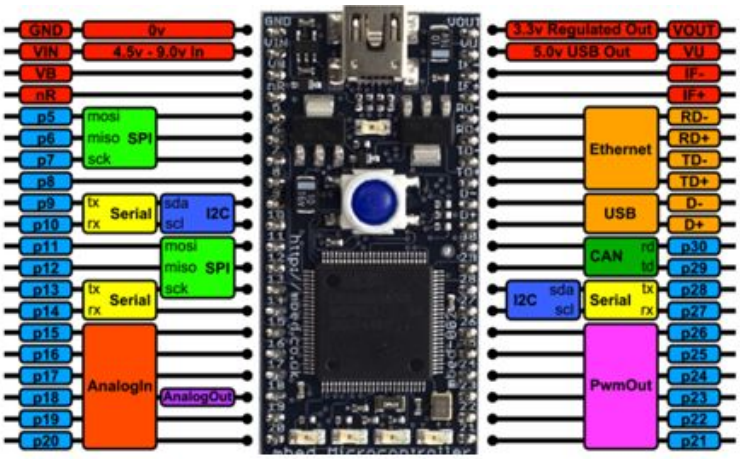
The following is a step by step guide on how to properly install hardware components, upload software to the microcontroller, and any necessary configurations that must be made.

- Step 1: First you will need to plug your Mbed NXP LPC1768 microcontroller into your application board like so:

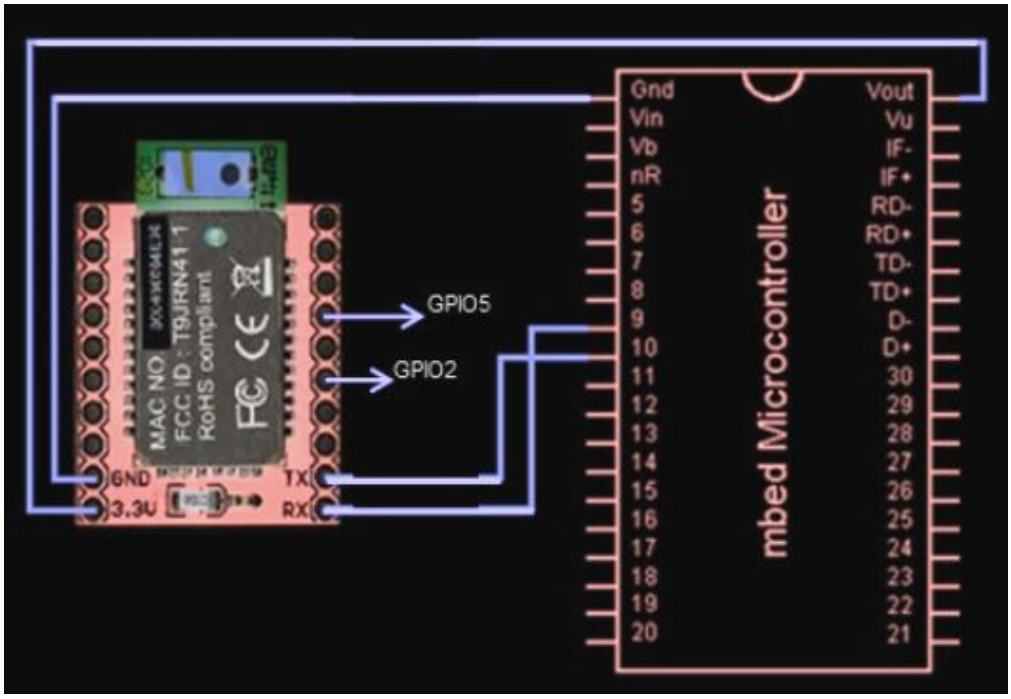


(Disregard the bottom sensor and all wiring for now)

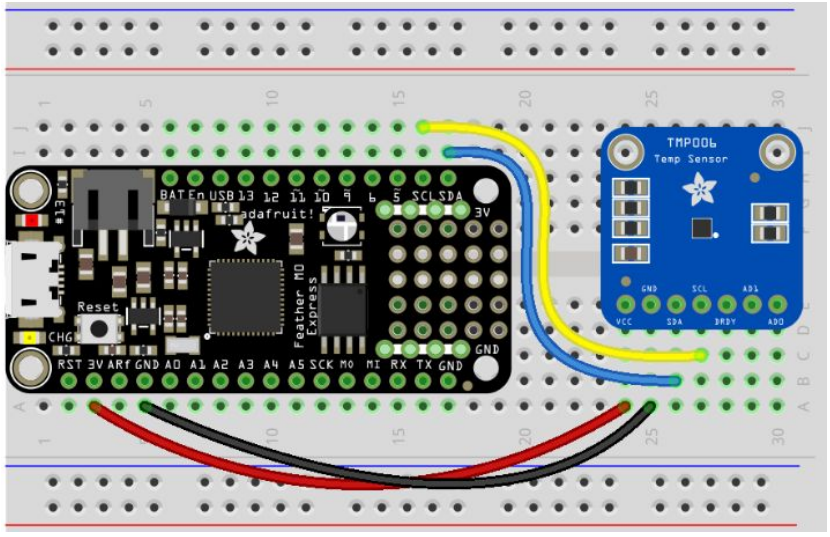
- Step 2: Take your bluetooth chip, either RN41 or RN42, and solder 4 wires on the bottom 2 pins on the both the left and right side of the chip. Once the wires are properly soldered, they need to be connected to the corresponding pins on the application board like in the image above. The connections need to correspond to the correct pins on the microcontroller. These pins include the GND pin, the Vout pin, and 2 serial communication pins that include TX and RX pin connections. There are 3 pairs of these connections located on the mbed microcontroller and any will do. You can see these connection locations in the following image.



Proper bluetooth chip connection will look something like this:



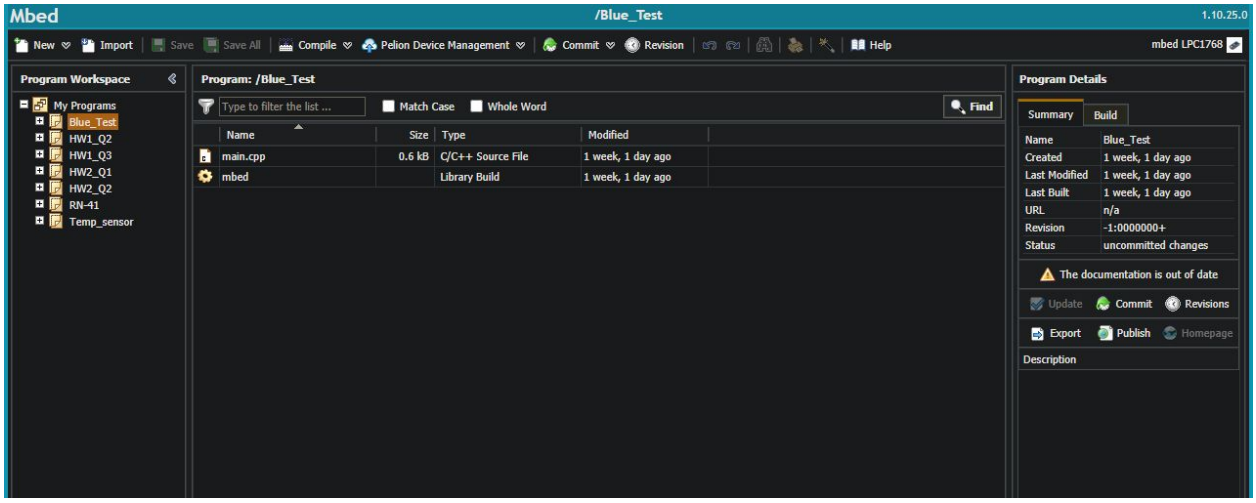
- Step 3: Repeat step 2 for the TMP006 temperature sensor. The same serial connections will be used, just pick a different pair of connections located on the microcontroller. the same GND and Vout pins will also need to be connected as well. It is virtually the same exact process as step 2. Again, it should look very similar to the following image:



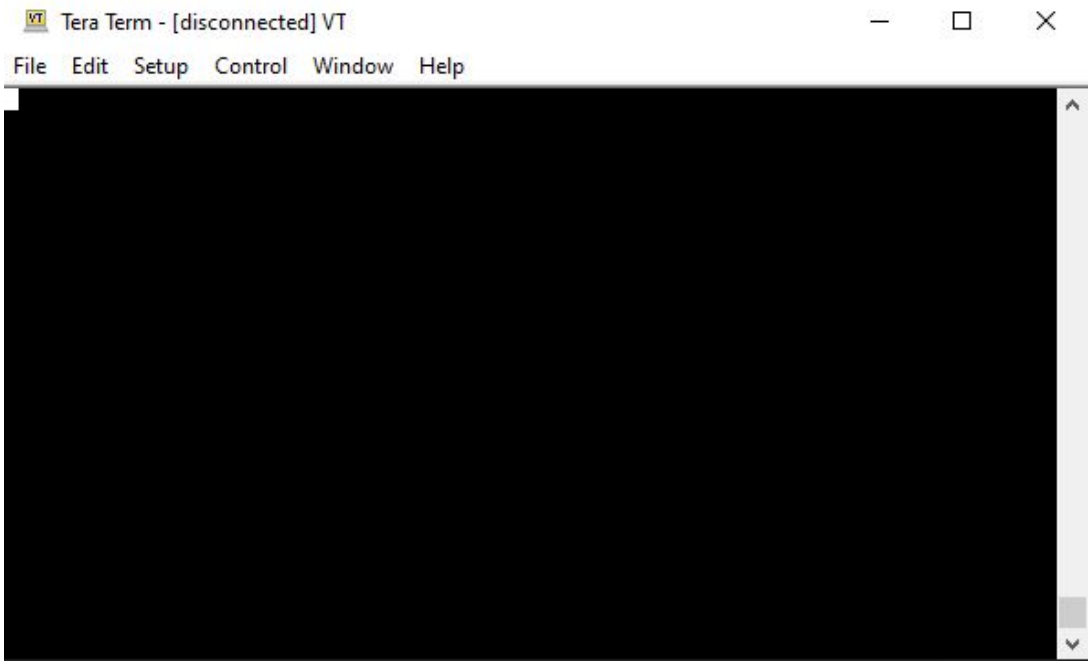
- Step 4: Connect a mini-USB cable to the serial connection located on the Mbed microcontroller and plug it into a PC for software upload and configuration. It is the only connection located on the board and can be seen here:

- Step 5: If all connections were completed successfully and USB cable is properly connected to a source of power (i.e. the host pc) then a single blue led light should light up on the microcontroller.

- Step 6: Register the board on Mbed's website that can be found in the board packaging. After making an account and registering your board, either download the IDE with the mbed compiler on it, or use Mbed's online compiler. For a more streamlined approach, the online compiler is recommended. It looks like this:

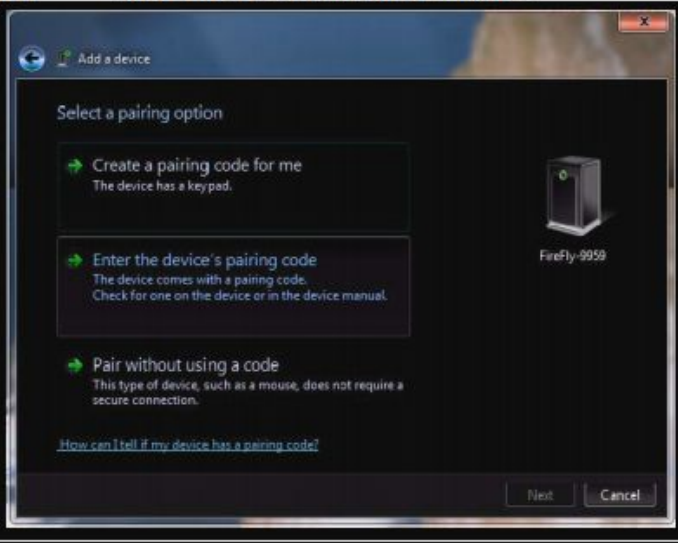


- Step 7: Download Tera Term or Putty (if you are using a Mac). It will look like this:



- Step 8: Pair your bluetooth chip connection with your PC and check which COM ports are incoming and outgoing. It will look like this:

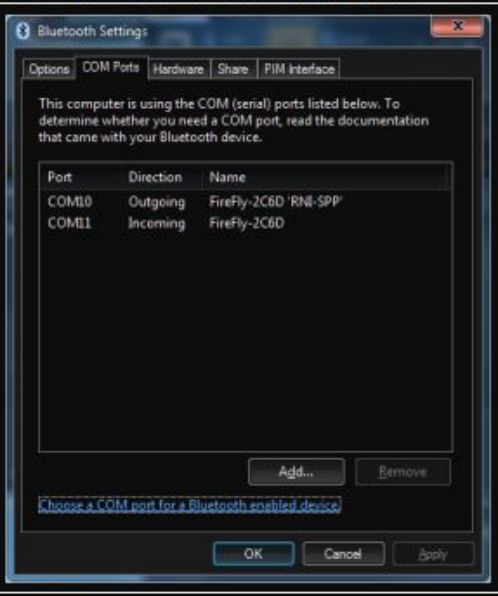
FIGURE 1-4: PAIR WITH THE BLUETOOTH MODULE



Note: You only need to pair with the module once.

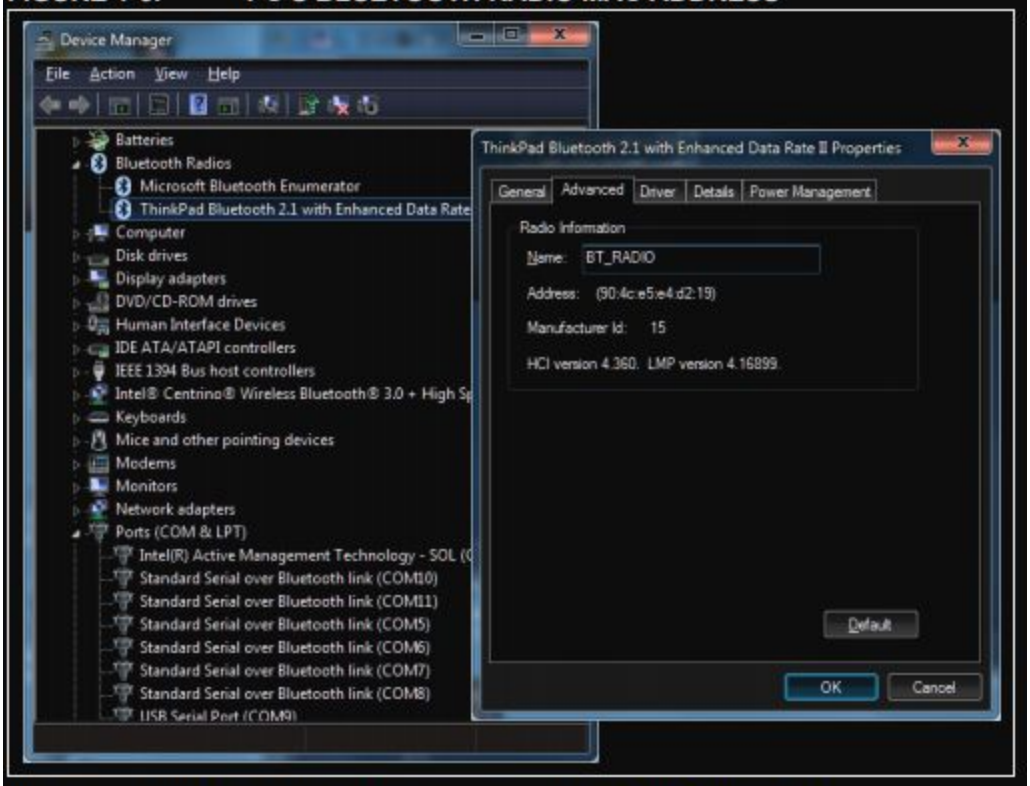
Figure 1-5 shows example COM port settings.

FIGURE 1-5: BLUETOOTH COM PORT SETTINGS



Step 9 : Following Roving Networks RN documentation, follow these steps to configure your bluetooth chip properly:

FIGURE 1-6: PC'S BLUETOOTH RADIO MAC ADDRESS



2. Pair your module with the PC as described in "Pairing" on page 13.
3. Open a terminal (called terminal A in this example) and connect it to the module. You can run this terminal on the host PC or another computer.
4. Open a second terminal (called terminal B in this example) on the host PC to listen for the incoming Bluetooth connection using the incoming COM port number.
5. Type `C,`*<MAC address>* <cr> in terminal A to establish an SPP connection to the host PC. See Figure 1-7 for an example connection.
6. Try the following commands:
   - `$$$` to enter command mode
   - `SO,%` to enable status message to see connect/disconnect conditions
   - `R,1` to reboot
   - `$$$` to re-enter command mode
   - `+` to enable local echo
   - `C,`*<MAC address>* to attempt a connection with remote device

   Characters you type in terminal B are sent over Bluetooth to the host PC and appear in terminal A. Any characters entered in terminal A are transmitted to terminal B.
7. To kill the connection, type the `k,1` <cr> command in terminal B.

- Step 9: If all steps until now have been successfully completed, you are now ready to begin uploading whatever software you wish to the mbed controller. There is an API available for the bluetooth hardware components and is available here:
  - https://os.mbed.com/users/NoLiver92/code/RN41/

- Step 10: Test your hardware/software and enjoy developing fun and exciting new ways to be creative with the Mbed suite!

7 Appendices

Be careful when configuring the RN41/42 bluetooth chip. Some of the newer models of this chip have a mini-USB port that allows for reconfiguration of the chip via serial connection. However, on the older models the only way to configure the chip settings is via bluetooth connection and if the chip is put into any of the master modes available to configure, you will not be able to re-enter the command mode which is the only way you can configure any settings. The only way to fix your chip if this happens is to solder a wire to the GP104 pin and hard factory reset it this way.