CG2271: Real-Time Operating Systems

USRTwBT

We will now explore how we can use the UART module in the FRDM board to communicate wirelessly using the BT module (BT06). The full video explanation and demo can be found in these links:

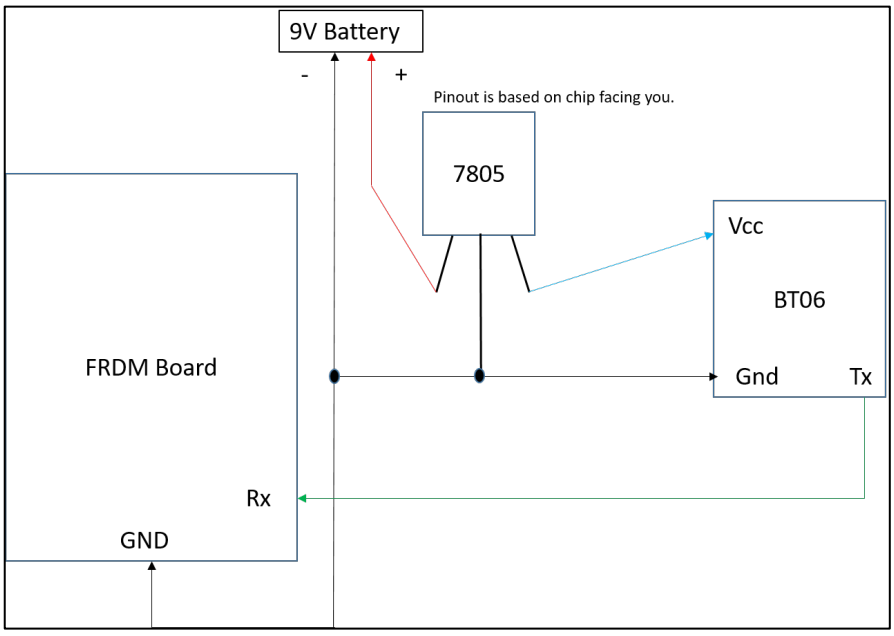
Detailed Explanation: https://youtu.be/1oMc1ua-ZbY

Demo: https://youtu.be/mlgzqk4Z8Kc

The steps below follow the explanation in the video.

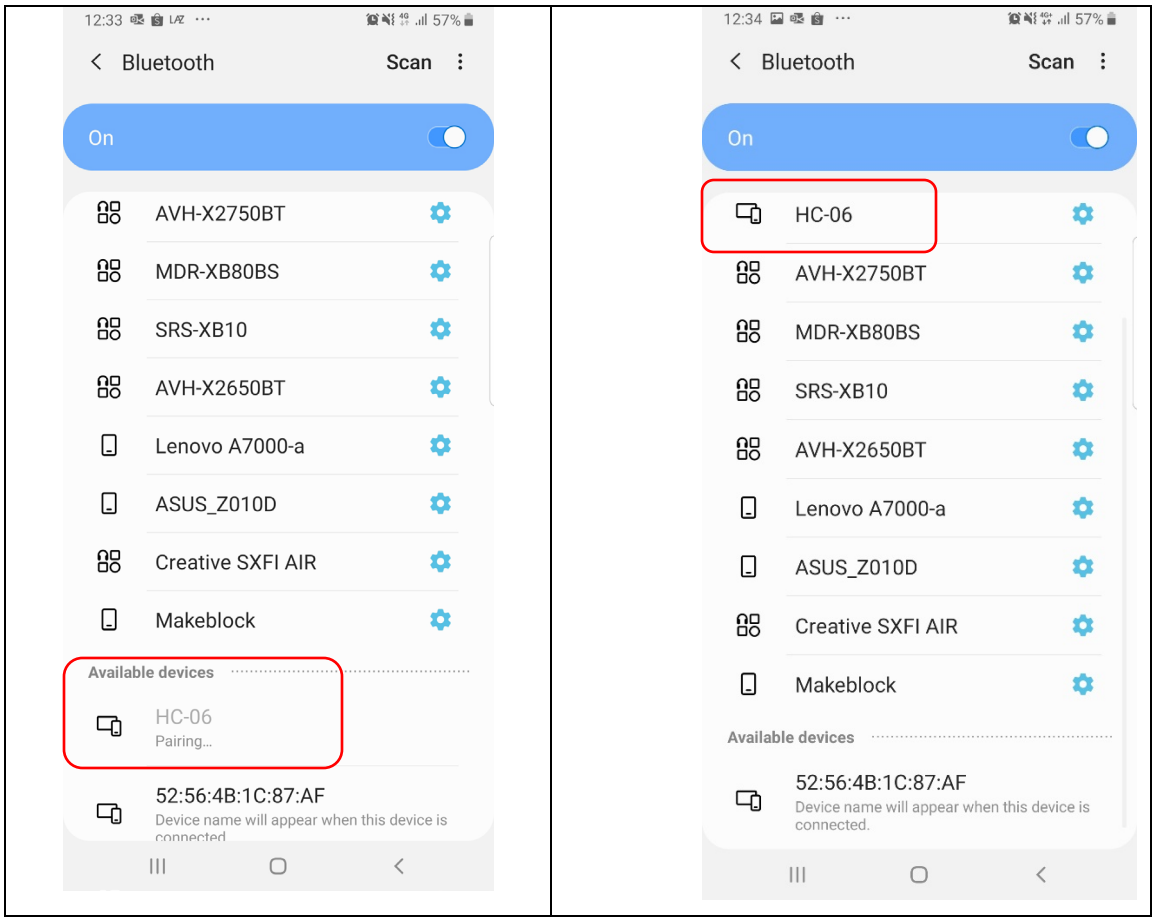
Step 1: Powering the BT06 module and connecting them up.

The BT06 requires 3.6V to 6V to operate. There is a 5V pin (Pin 16) on the FRDM board. However, the current is insufficient for the requirements of the BT module. As such, we will be using the 7805 Voltage Regulator powered with the 9V battery for the device. The connections are as shown below.



Step 2: Pairing the Phone with the BT06

When the BT06 is first powered up, you will see the red LED on it blinking. At this time, you need to activate the Bluetooth module on your phone and initiate a pairing with the BT device.



Step 3: Creating the Android App

You are allowed to use any App Development Platform for the App. The easiest way to do it would be to use the AppInventor platform here: https://appinventor.mit.edu/

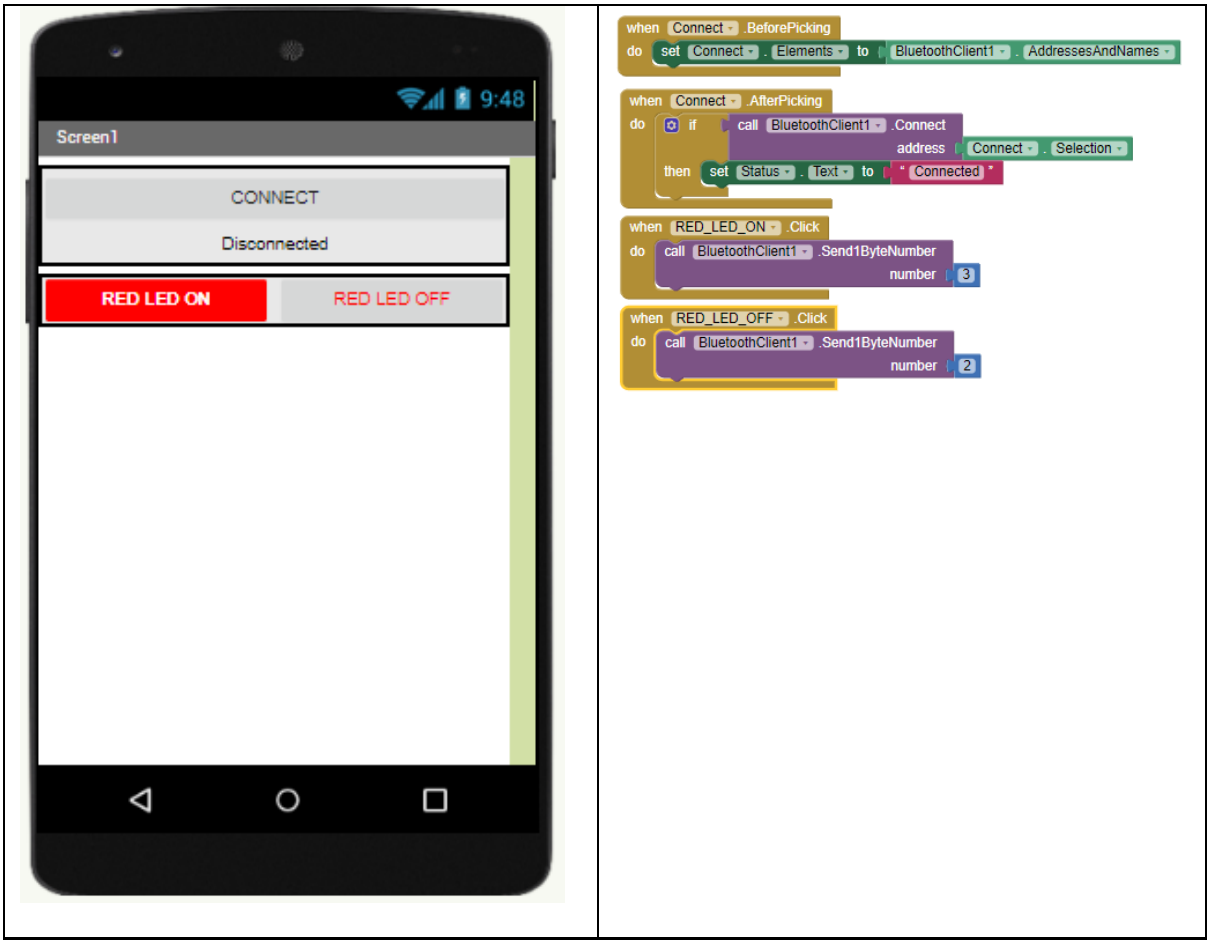
The key points to take note are:

CONNECT Button: ListPicker

Connect/Disconnected: Label

Control Buttons like Move Motor / Light LED, etc: Button

A basic app and the screenshot of the functions used are shown below.



Step 3: Data Packet Format

The data packet format is up to you to decide. You can choose to encode all the information that you need to send over within a single byte, or across multiple bytes. A single byte data packet will be easier to handle by the Receiver. The video gives you an idea on how to use the various bits for different commands.

Step 4: Receive the Data Packet

The FRDM board RX module must be configured to receive the data and decode it. The Tx and Rx functionalities of the microcontroller were covered in the Lecture. During the Lab, you will explore it in more detail. That should give you the knowledge you need to complete the code with the Polling approach. A snapshot of the specific code sections are shown below.

```
 8  #define BAUD_RATE 9600
 9  #define UART_TX_PORTE22 22
10  #define UART_RX_PORTE23 23
11  #define UART2_INT_PRIO  128
12
13  #define LED_RED 2 // 0b00000010
14  #define LED_MASK(x)   (x & 0x06)
15  #define BIT0_MASK(x) (x & 0x01)
16
17  #define RED_LED   18  // PortB Pin 18
18  #define GREEN_LED 19  // PortB Pin 19
19  #define BLUE_LED  1   // PortD Pin 1
20  #define MASK(x) (1 << (x))
```

```
156  /* MAIN function */
157  int main(void)
158  {
159    uint8_t rx_data = 0x01;
160
161    SystemCoreClockUpdate();
162    initUART2(BAUD_RATE);
163    InitGPIO();
164    offRGB();
165    while(1)
166    {
167      /* Rx and Tx*/
168      rx_data = UART2_Receive_Poll();
169
170      if(LED_MASK(rx_data) == LED_RED)
171      {
172        if(BIT0_MASK(rx_data))
173          ledControl(led_mapping[0][1], led_on);
174        else
175          ledControl(led_mapping[0][1], led_off);
176      }
177    }
```

The rest of the UART code will be covered in the Lab. The LED control is what you had done earlier for the GPIO lab. This example is specific to the control of LED's only. You need to develop your own data packet format in order to fulfil all the requirements of the project.

Thank You!

Good Luck! ☺