Lab 10: Bluetooth Communication

Name: Shahmir Mustafa Chaudhry ID: 08400` Section: T1

Objective

To build wireless communication channel between TivaC TM4C123 and Android device using Bluetooth module HC-05

In-Lab

Task 1: Establish Bluetooth communication between TivaC and Android to transmit and receive characters

Task 2: To display different colors on RGB LED and control servo using Bluetooth

Task 3: LED Control with HC-05 and Andriod App using Keil uvision

1 Bluetooth Module HC-05

Wireless communication is swiftly replacing the wired connection when it comes to electronics and communication. Designed to replace cable connections HC-05 uses serial communication to communicate with the electronics. Usually, it is used to connect small devices like mobile phones using a short-range wireless connection to exchange files. It uses the 2.45GHz frequency band. The transfer rate of the data can vary up to 1Mbps and is in range of 10 meters. The HC-05 module can be operated within 4-6V of power supply. It supports baud rate of 9600, 19200, 38400, 57600, etc.

This module can be used to perform low-cost wireless communication between microcontrollers such as Arduino, TM4C123, Pic Microcontroller, Raspberry Pi, BeagleBone, STM32, and Arduino Mega, etc. Also, we can use it for communication between a microcontroller and PC or Android application. There are many Android applications available in the Google Play store which we can use to send and receive data to/from HC05. The pin configuration for this module is given in Fig. 7 with description given below.

- Enable (Key) - This pin is used to set the Data Mode or and AT command mode (set high)
- VCC - This is connected to +5V or +3.3V power supply

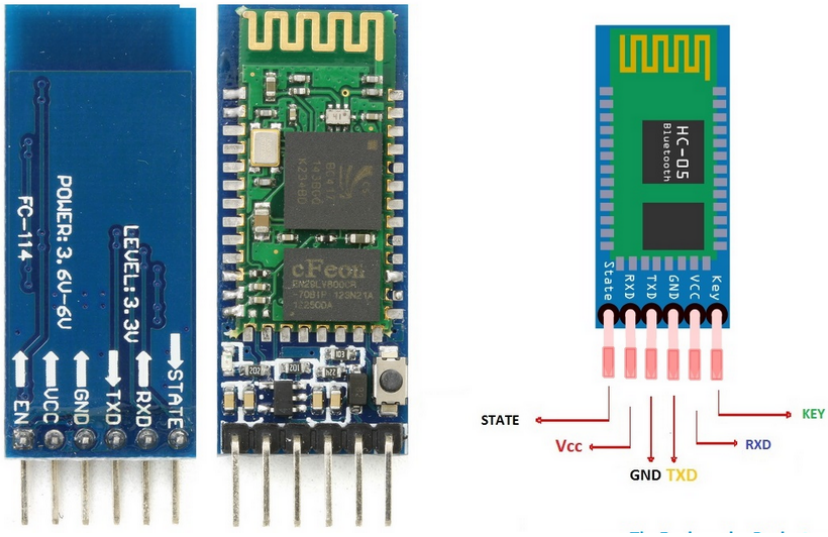


Figure 1: Pin Configuration for HC-05

- Ground - Connected to ground of powering system
- Tx (Transmitter) - This pin transmits the received data Serially
- Rx (Receiver) - Used for broadcasting data serially over Bluetooth
- State - Used to check if the Bluetooth is working properly

2 UART communication in TM4C123GH6PM

UART stands for Universal Asynchronous Receiver/Transmitter. It's not a communication protocol like SPI and I2C, but a physical circuit in a microcontroller, or a stand-alone IC. A UART's main purpose is to transmit and receive serial data using just two wire TX and RX.

UART Module	Rx Pin	Tx Pin
UART0	PA0	PA1
UART1	PC4	PC5
UART2	PD6	PD7
UART3	PC6	PC7
UART4	PC4	PC5
UART5	PE4	PE5
UART6	PD4	PD5
UART7	PE0	PE1

Figure 2: UART Channels in TM4C123GH6PM

HC-05 is a two-way full-duplex Bluetooth to serial module. This Bluetooth device operates on UART communication. In other words, it uses serial communication to transmit and receive data serially over standard Bluetooth radio frequency.

UART communication port of the TM4C123GH6PM microcontroller is used with HC-05. This microcontroller has on-chip 5 UART ports such as UART0 to UART5. We can use any one of these UART ports to interface TM4C123 with the BT device. The table in Fig. 2 shows all UART ports of TM4C123 Tiva launchpad and their corresponding GPIO pins.

3 Android Bluetooth Terminal App

In order to establish Bluetooth communication between TivaC and Android device, we will use a android application named ”Serial Bluetooth Terminal”, freely available on Play Store app. You are required to download and install that application. Once the application would be downloaded, and our HC-05 connections are made with TivaC in Task 01 below, we will follow the steps shown in Fig. 3 to send and receive commands over Serial monitor of the app and the Energia.

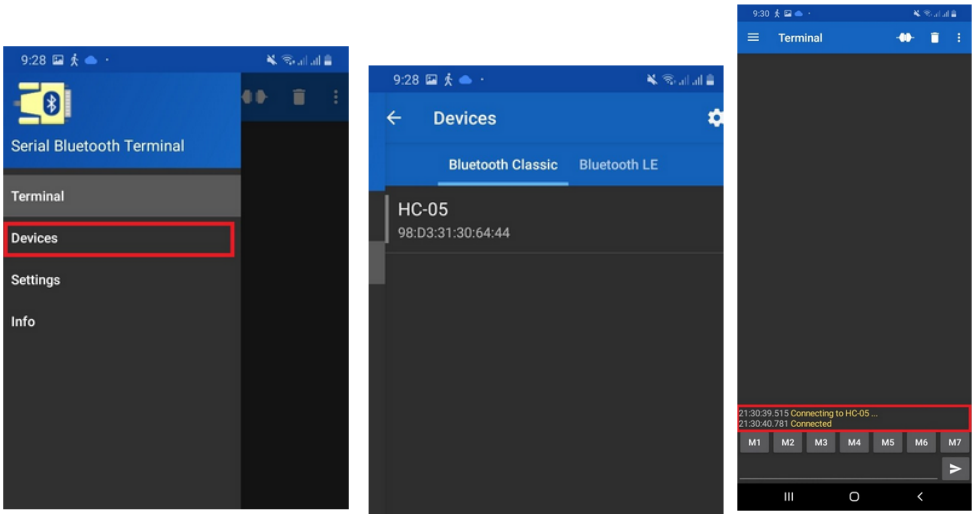


Figure 3: Steps for Serial Bluetooth Terminal App

4 TivaC LaunchPad with Energia

In Energia IDE, unlike Kiel uVision, we can use TivaC LaunchPad pins for various peripherals without the need to activate Ports using registers or specifying function of the pin. But it also comes with limitations of usability and programmable scope of the board. In Energia, we can refer to Pins of TivaC directly using numeric digit like 1,2,3.. and so on. Pin map for the EK-TM4C123GXL LaunchPad is given in Fig. 4 with Black Columns under J1, J2, J3 and J4.

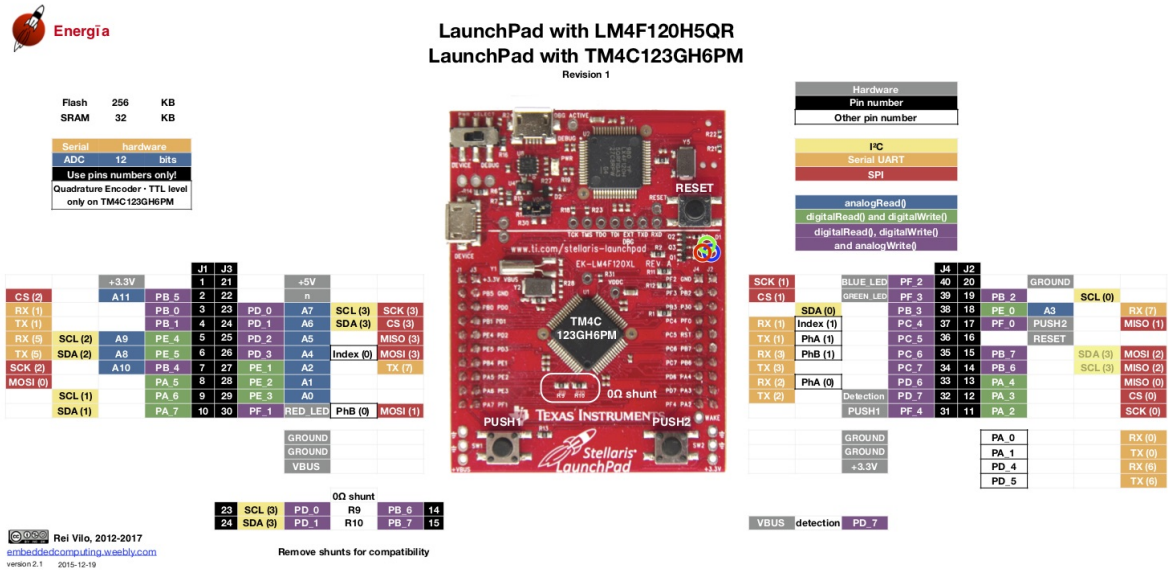


Figure 4: TivaC Pin Distribution with Energia

In-Lab Tasks

Task 1: Establish Bluetooth communication between TivaC and Android to transmit and receive characters

In this task, we are going to establish simple Bluetooth communication using UART0 of TivaC and send characters and string from Android device to be displayed on Energia Serial Monitor. Follow the step below to achieve the communication:

- Make the connections given in Fig. 5.
- Download and open ”bluetooth.ino” in Energia.
- Modify the code where you are required to in order to make it functional.
- Download and Install ”Serial Bluetooth Terminal” in your Android Mobile Device.
- Connect your mobile device with Bluetooth module using steps given in Fig. 3 with password ”1234” or ”0000”. The Bluetooth device name is mentioned on module itself.
- Once a successful communication is established, you should be able to send and receive the characters using Serial.

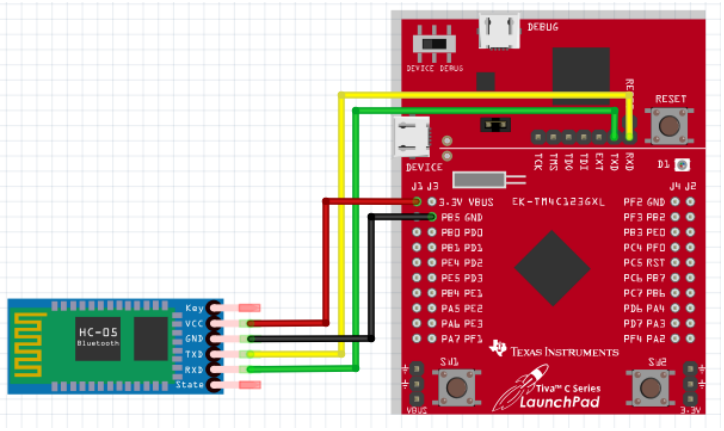


Figure 5: Circuit Connection between TivaC and HC-05

Write your brief and complete understanding of how the code works. Explain how Bluetooth module use UART protocol to communicate?

The code controls a servo motor through Bluetooth commands received from a paired device, such as a smartphone, using a Bluetooth module like the HC-05.
- It sets up UART communication, configuring UART5 on the TivaC microcontroller at a baud rate of 9600 to match the Bluetooth module's rate.
-When a command is sent from the Bluetooth device, it is received by the TivaC over the UART connection, which enables asynchronous data exchange without needing a shared clock.
-Each received command corresponds to a specific angle for the servo motor—for example, a command like 'A' might move the servo to 45 degrees, while 'B' moves it to 90 degrees.
-After executing the command, the TivaC can send a confirmation message back to the Bluetooth device, giving feedback that the action was
-This setup provides a convenient way to wirelessly control the servo motor's position through simple commands.

Provide your clear circuit image below.

Attached in comments

Provide your code here with appropriate comments below (Get the circuit demonstration checked with RA within Lab)

```
#include <Servo.h>
Servo myservo;  // Creating a Servo Object
int pos = 0;    // Position of Servo
const int servoPin = 26; // Pin at PD3

// Define the RGB pins
const int red = 30;     // Present at PF1
const int green = 40;   // Present at PF2
const int blue = 39;    // Present at PF3

int colorDetect = 0;     // For different color detection
char received;           // Characters to be send/receive on UART

void setup() {
  // Initialize Baud Rate at 9600
  Serial.begin(9600);

  // Specify pin modes for red, green, and blue pins
  pinMode(red, OUTPUT);   // Red color PWM pin defined as output
  pinMode(green, OUTPUT); // Green color PWM pin defined as output
  pinMode(blue, OUTPUT);  // Blue color PWM pin defined as output

  // Start with no color on RGB LEDs
  analogWrite(red, 0);
  analogWrite(green, 0);
  analogWrite(blue, 0);

  // Attach servo to specified pin
  myservo.attach(servoPin);
}

void loop() {
  // Check if data is available from Bluetooth terminal
  if (Serial.available() > 0) {
    // Read Bluetooth data and store it in colorDetect variable
    colorDetect = Serial.read();

    // Print received character for debugging
    char Rec = char(colorDetect);
    if (Rec != '0') {
      Serial.println(Rec); // Visualize the received character on the serial monitor
    }
  }

  // Call chooseColor function with colorDetect as argument
  chooseColor(colorDetect);
}

void chooseColor(char color) {
//  // Set colors and servo position based on received character
//  switch (color) {
//    case 'R': // Red color
//      analogWrite(red, 255);
//      analogWrite(green, 0);
//      analogWrite(blue, 0);
//      myservo.write(0); // Set servo to position 0
//      break;
//
//    case 'G': // Green color
//      analogWrite(red, 0);
//      analogWrite(green, 255);
//      analogWrite(blue, 0);
//      myservo.write(45); // Set servo to position 45
//      break;
//
//    case 'B': // Blue color
//      analogWrite(red, 0);
//      analogWrite(green, 0);
//      analogWrite(blue, 255);
//      myservo.write(90); // Set servo to position 90
//      break;
//
//    case 'W': // White color
//      analogWrite(red, 255);
//      analogWrite(green, 255);
//      analogWrite(blue, 255);
//      myservo.write(135); // Set servo to position 135
//      break;
//
//    case 'Y': // Yellow color
//      analogWrite(red, 255);
//      analogWrite(green, 204);
//      analogWrite(blue, 0);
//      myservo.write(180); // Set servo to position 180
//      break;
//
//    default: // LEDs off
//      analogWrite(red, 0);
//      analogWrite(green, 0);
//      analogWrite(blue, 0);
//      myservo.write(90); // Set servo to neutral position
//      break;
//  }
}
```

Task 2: To display different colors on RGB LED and control servo using Bluetooth

In this task, we will control the color of the built-in RGB LED of TivaC and position of servo according to command received by Android Terminal. In order to achieve the above, follow the instructions below:

- Connect Servo's VCC, GND and PWM pins to 3.3V/Bus, GND and PD3 of the TivaC Board.
- Initialize pinModes for Servo PWM, and built-in RGB LED which is present on PortF at pins PF1, PF2 and PF3.
- Use myservo.write(x) to rotate servo to position x.
- Use analogWrite() to obtain different colors on built-in RGB LED.

Using the above steps, you are required to achieve the following in this task:

- Servo to be moved 45 degrees with color RGB color RED on command "A"
- Servo to be moved 90 degrees with color RGB color GREEN on command "B"
- Servo to be moved 135 degrees with color RGB color BLUE on command "C"
- Servo to be moved 180 degrees with color RGB color WHITE on command "D"

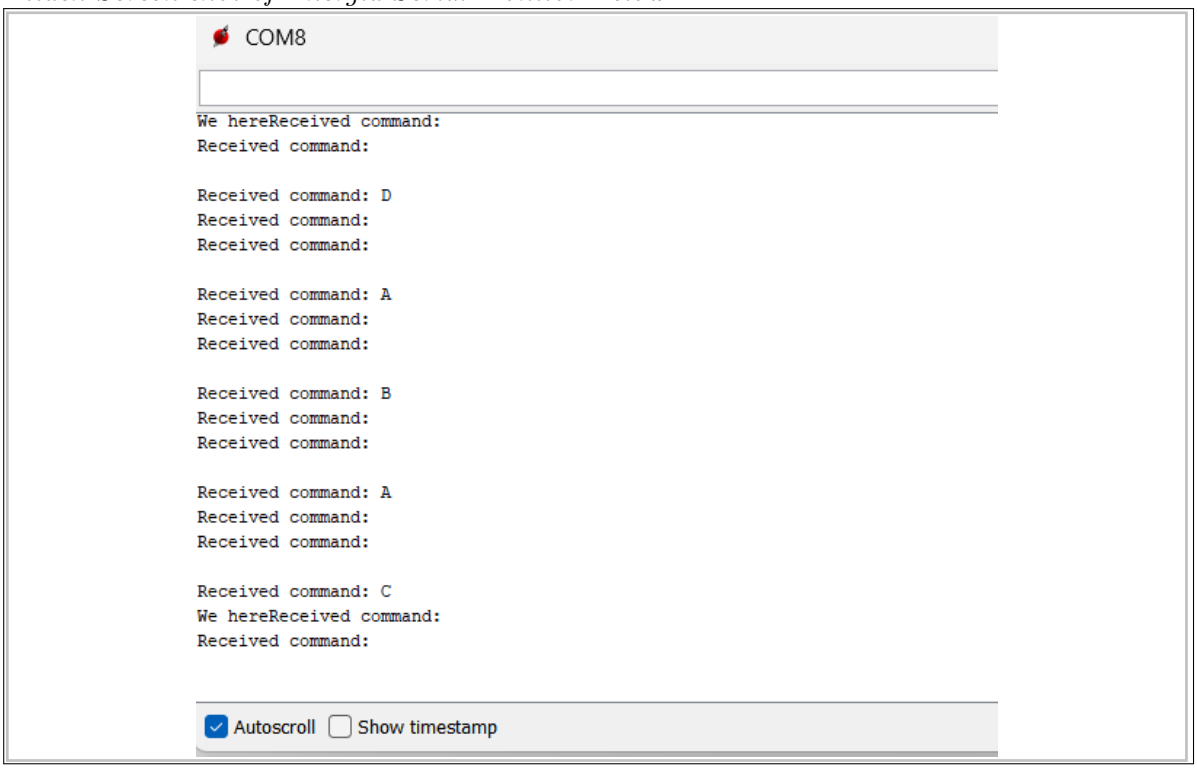
Write your brief and complete understanding of how the code works for controlling servo and RGB color using Bluetooth communication.

-The code allows for wireless control of an RGB LED and a servo motor on the TivaC microcontroller through Bluetooth communication.
-Using an HC-05 Bluetooth module, commands sent from a Bluetooth-enabled device, such as a smartphone, are received and processed by the TivaC.
-Each command is a character that triggers a specific action: adjusting the RGB LED color or moving the servo motor to a preset angle.
-The RGB LED, connected to GPIO pins, can display various colors by turning the red, green, and blue channels on or off in different combinations.
-The servo motor, connected to a designated pin, is moved to specific angles based on received commands.
-After executing each command, the microcontroller can send feedback to the Bluetooth device, confirming the action taken. This setup allows for convenient remote control of both the LED and the servo using a simple Bluetooth interface.

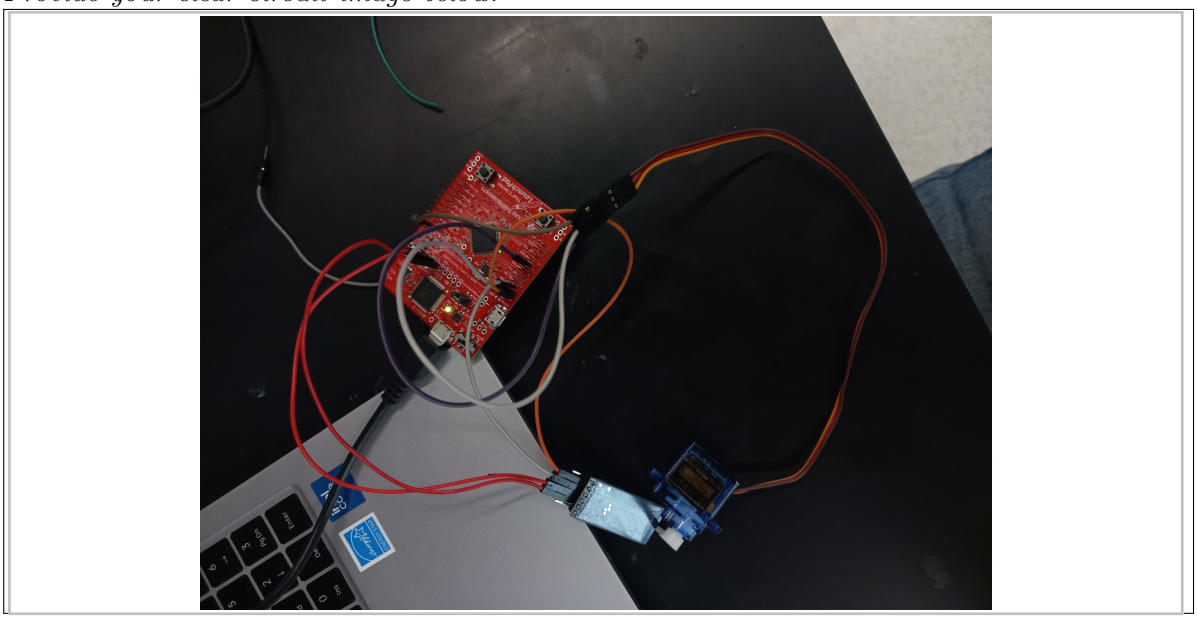
Provide your code here with appropriate comments below (Get the circuit demonstration checked with RA within Lab)

```
#include <Servo.h>
Servo myservo;  // Creating a Servo Object
const int servoPin = 26; // Pin at PD3

// Define the RGB pins on TivaC
const int red = 30;     // PF1
const int green = 40;   // PF2
const int blue = 39;    // PF3

char colorDetect = 0;   // For different color detection

void setup() {
  // Initialize Baud Rate at 9600
  Serial.begin(9600);

  // Specify pin modes for RGB pins
  pinMode(red, OUTPUT);
  pinMode(green, OUTPUT);
  pinMode(blue, OUTPUT);

  // Start with no color on RGB LEDs
  analogWrite(red, 0);
  analogWrite(green, 0);
  analogWrite(blue, 0);

  // Attach servo to specified pin
  myservo.attach(servoPin);
}

void loop() {
  // Check if data is available from Bluetooth terminal
  if (Serial.available() > 0) {
    // Read Bluetooth data and store it in colorDetect variable
    colorDetect = Serial.read();

    // Print received character for debugging
    if (colorDetect != '0') {
      Serial.print("Received command: ");
      Serial.println(colorDetect);
    }
  }

  // Call chooseColor function with colorDetect as argument
  chooseColor(colorDetect);
}

void chooseColor(char color) {
  // Set colors and servo position based on received character
  switch (color) {
    case 'A': // Red color, 45 degrees
      analogWrite(red, 255);
      analogWrite(green, 0);
      analogWrite(blue, 0);
      myservo.write(45);
      break;

    case 'B': // Green color, 90 degrees
      analogWrite(red, 0);
      analogWrite(green, 255);
      analogWrite(blue, 0);
      myservo.write(90);
      break;

    case 'C': // Blue color, 135 degrees
      analogWrite(red, 0);
      analogWrite(green, 0);
      analogWrite(blue, 255);
      myservo.write(135);
      Serial.print("We here");
      break;

    case 'D': // White color, 180 degrees
      analogWrite(red, 255);
      analogWrite(green, 255);
      analogWrite(blue, 255);
      myservo.write(180);

      break;

    default: // LEDs off, neutral servo position
      analogWrite(red, 0);
      analogWrite(green, 0);
      analogWrite(blue, 0);
      myservo.write(90); // Neutral position
      break;
  }
}
```

Attach Screen-shot of Energia Serial Monitor Below



Provide your clear circuit image below.



Task 3: LED Control with HC-05 and Andriod App using Keil uvision

TM4C123 Tiva Launchpad has an RGB LED connected with PF1, PF2, and PF3 pins. We will send ON/OFF commands from an Android application. Based on these commands, LEDs will turn on or off.

Commands used to control onboard LEDs of Tiva Launchpad are listed below:

Command From App	Function
A	Green LED ON
B	Green LED OFF
C	RED LED ON
D	RED LED OFF
E	BLUE LED ON
F	BLUE LED OFF

Figure 6: Commands for LEDs

For example, if we send the character 'A' from the Android app to HC-05 the green LED of Tiva Launchpad will turn on. Similarly, if we transmit 'B' from the Android app to HC-05 green LED of Tiva Launchpad will turn off. Likewise, commands for the other two LEDs are mentioned in the above table.

- We will use a UART5 port. For the UART5 port, the PE4 GPIO pin is a receiver pin (Rx) and the PE5 pin is a transmitter pin (Tx). Now make connections with HC-05 Bluetooth module and Tiva Launchpad according to schematic diagram (Figure 7).
- Now create a new project in Keil uvision and compile the skeleton code (will be given in modules section on LMS). After that upload the generated hex file to Tiva Launchpad.
- Modify the code where you are required to in order to make it functional.
- Download and Install "Serial Bluetooth Terminal" in your Android Mobile Device.
- Connect your mobile device with Bluetooth module using steps given in Fig. 3 with password "1234" or "0000". The Bluetooth device name is mentioned on module itself.
- Once a successful communication is established, you should be able to send and receive the characters using Serial.

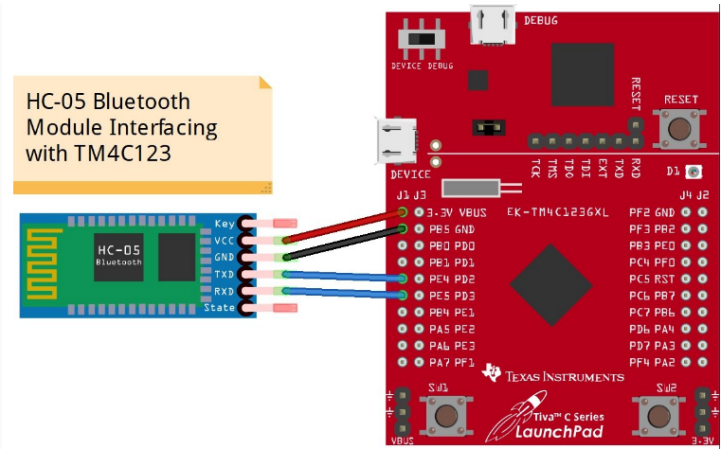


Figure 7: Circuit Connection

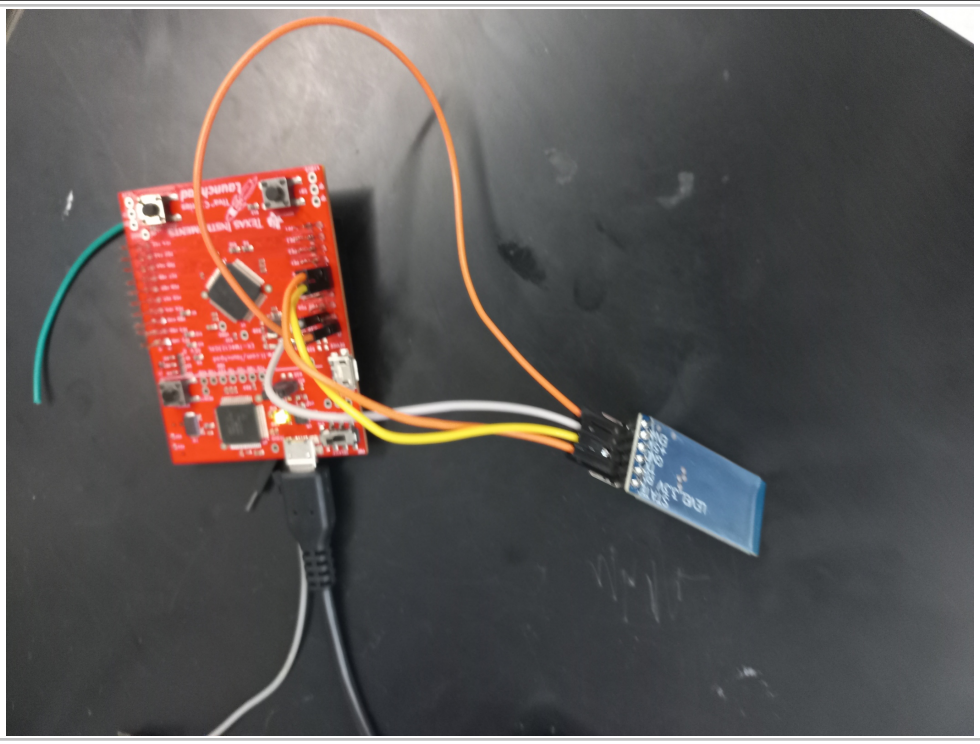
Write your brief and complete understanding of how the code works.

- turn off all leds
- turn off the system define clock since it works seperate in this scenario
- the receiver receives the signal from the transmitter of the bluetooth module
-the tiva c reads and impements this signal command
-the transmitter PE5 transmit the message to the bluetooth receiver
- the bluetooth module receives the meessage and displays it in the terminal
- the bluetooth module behaves as a communication protocol effective transmitting commands to the tiva c board

Provide your code here with appropriate comments below (Get the circuit demonstration checked with RA within Lab)

```
#include "TM4C123.h"
#include <stdint.h>
#include <stdlib.h>

void Delay(unsigned long counter);            // Used to add delay
void HC05_init(void);                         // Initialize UART5 module for HC-05
char Bluetooth_Read(void);                    // Read data from Rx5 pin of TM4C123
void Bluetooth_Write(unsigned char data);     // Transmit a character to HC-05 over Tx5
void Bluetooth_Write_String(char *str);       // Transmit a string to HC-05 over Tx5

int main(void) {
  HC05_init(); // Initialize UART5 of TM4C123GH6PM for HC-05

  // Enable clock to GPIOF and set PF1, PF2, PF3 as digital output pins
  SYSCTL->RCGCGPIO |= 0x20;         // Enable clock to GPIOF
  GPIOF->DIR |= 0x0E;               // Set PF1, PF2, and PF3 as output
  GPIOF->DEN |= 0x0E;               // Enable digital function for PF1, PF2, and PF3
 GPIOF->DATA &= 0;
 Delay(5);

  while(1) {
    char c = Bluetooth_Read(); // Get a character from UART5

    // Check the received character and control the RGB LED
    if (c == 'A') { // Green LED ON
      GPIOF->DATA |= (1 << 3);          // Turn on PF3 (green LED)
      Bluetooth_Write_String("Green LED ON\n");
    }
    else if (c == 'B') { // Green LED OFF
      GPIOF->DATA &= ~(1 << 3);         // Turn off PF3 (green LED)
      Bluetooth_Write_String("Green LED OFF\n");
    }
    else if (c == 'C') { // Blue LED ON
      GPIOF->DATA |= (1 << 2);          // Turn on PF2 (blue LED)
      Bluetooth_Write_String("Blue LED ON\n");
    }
    else if (c == 'D') { // Blue LED OFF
      GPIOF->DATA &= ~(1 << 2);         // Turn off PF2 (blue LED)
      Bluetooth_Write_String("Blue LED OFF\n");
    }
    else if (c == 'E') { // Red LED ON
      GPIOF->DATA |= (1 << 1);          // Turn on PF1 (red LED)
      Bluetooth_Write_String("Red LED ON\n");
    }
    else if (c == 'F') { // Red LED OFF
      GPIOF->DATA &= ~(1 << 1);         // Turn off PF1 (red LED)
      Bluetooth_Write_String("Red LED OFF\n");
    }
   }
}

// Initialization of UART5 for HC-05 Bluetooth communication
void HC05_init(void) {
  SYSCTL->RCGCUART |= 0x20;         // Enable clock to UART5
  SYSCTL->RCGCGPIO |= 0x10;         // Enable clock to PORTE for PE4/Rx and PE5/Tx
  Delay(1);

  UART5->CTL = 0;                   // Disable UART5 module
  UART5->IBRD = 104;                // Integer portion of baud rate divisor
  UART5->FBRD = 11;                 // Fractional portion of baud rate divisor
```

Provide your clear circuit image below.



5 Assessment Rubrics

Marks distribution

		LR1	LR2	LR4	LR5	LR9
	Task 1	-	10 points	05 points	10 points	15 points
	Task 2	10 points	10 points	05 points	10 points	
In-lab	Task 3	-	10 points	05 points	10 points	
Total Marks 100						

Marks obtained

		LR1	LR2	LR4	LR5	LR9
	Task 1	-				
	Task 2					
In-lab	Task 3	-				
Total Marks 100						