Programming with the Intel Galileo

Bluetooth

Intro
Today we will learn how to communicate between an Android phone and the Intel Galileo via Bluetooth.

The Bluetooth module we will be using is the “HC-06” which acts as a serial port through which you can send and receive data using a terminal or a Bluetooth customized application on your computer or phone. In this lab we will be using an Android phone.

Getting Started

Start by downloading a terminal simulator application for your android phone(If you are in reach of an Android phone, an alternative would be to use a Bluetooth enabled computer). An app that is recommended for such use is “Blueterm” by pymasde.es on the Google Play Store. For the rest of the lesson the demos and examples will use Blueterm although it will look similar to any application you choose.
The Bluetooth module we’ll be using in this lesson is the HC-06.

The wiring is very straightforward:

VCC → 5V
GND → GND
TXD → RX (Pin 0)
RXD → TX (Pin 1)

**Setting up**

Wire the Bluetooth module and power up the Intel Galileo. You will need to first pair the Bluetooth module with your phone or device. Navigate to your Bluetooth settings and scan for nearby devices. You will see a device with the default name ‘HC-06’. Connect with the default passcode of ‘1234’.

Once that is setup, open Blueterm and connect to the module using the menu key and selecting “Connect device”.
Interfacing between your Computer, Galileo, and Android

Arduino can interface between devices using Serial Communication. To communicate between the Galileo and your computer, you would use `Serial` and its functions such as `Serial.begin(9600)`, `Serial.available()`, `Serial.read()` etc.

Now, to navigate between the Galileo and any external Serial devices such as the HC-06, you would use `Serial1` and its functions such as `Serial1.begin(9600)`, `Serial1.available()`, `Serial1.read()` etc.

Use the following code to start. Upload it to the Galileo and make sure you are connected to the Bluetooth module with your Android phone. Start the serial monitor in the Arduino IDE.

```c
char value;

void setup() {
  Serial.begin(9600);
  Serial1.begin(9600);
}

void loop() {
  if(Serial1.available()){
    value = Serial1.read();
    Serial1.println(value);
    Serial.println(value);
  }
}
```

Start by typing characters on the Keyboard in Blueterm.
You’ll notice characters repeat on Blueterm and on the serial monitor they are displayed only once. That is because “local echo” is enabled in the Blueterm’s preferences. Depending on the exercise or project, it is necessary for it to be enabled. In this case, it’s preference.

Reading through the code above, the Gaileo first checks if data is available. If it is, the data is transmitted and saved into a char data type. The char is then printed on both the local Serial Monitor and Blueterm on the Android device.

This demonstrates the use of Serial and Serial1 each distinct between the Serial Monitor and Blueterm on the Android device.

Video Demonstration: [https://youtu.be/p8OorlsXf5o](https://youtu.be/p8OorlsXf5o)

**Exercise 1 - Bluetooth Chat**

Create a chat system that communicates between the Intel Galileo (using Serial Monitor) and the Android phone. (Watch video demo for examples on formatting)

**Criteria:**
- Every message should contain a prepended header that tells where the message is coming from and who it’s going to. For Example (“Galileo -> BT: “)
- When the return key is pressed on the Serial Monitor’s text box, the message should send and be displayed on both devices.
- When the return key is sent from the Android Device, the message should terminate and newlines should print accordingly.
- Format should be identical with video demo

**Hints:**
- From the Serial Monitor, if Serial.read() is called and there is nothing to be read, the function will return ‘-1’.
- Determine whether “local echo” in Blueterm is necessary.
- One newline in Blueterm will cause the app to resume at the next vertical position but not leftmost horizontal. Two newlines will start at the leftmost area of the screen.

**Exercise 2 – LED Control**

Control the state of one Blue LED and one Green LED. Using Bluetooth serial communication, a ‘B’ turns on the Blue LED, and a ‘b’ turns it off. A ‘G’ would turn on the green LED, and a ‘g’ turns it off.
Exercise 3 – RGB LED Control

Extend Exercise 2 by using an RGB LED instead of two LEDs where ‘R’ and ‘r’ controls RED, ‘B’ and ‘b’ controls BLUE, and ‘G’ and ‘g’ controls the GREEN pin of the RGB. Add a function for when ‘o’ is sent, all pins are set to LOW. Combine LEDs to combine colors.

Exercise 4 – Stepper Motor Control

Start by wiring up the Stepper Motor. Extend Exercise 5 of the Stepper Motor Lab to be controlled by the Android Phone via Bluetooth where ‘+’ and ‘-’ corresponds to Clockwise and CCW respectfully, and ‘*’ resets to its default position. Display the rotation information only in the Serial Monitor.

Hints:

- To simplify the transition, focus on which Serial calls need to be switched to Serial1 calls.