Supplementary Arduino Programming and App Development Laboratory Information

Arduino Programming

Bluetooth Motor Control For Older ANDROID devices (Regular bluetooth NOT LE)

This program will use incoming data from a Bluetooth compatible device to control the motors with the Bluetooth chip BLUESMiRF. The sensor will act as a check that will limit when the motors can be used.

1. Open a new sketch and include the AFMotor library. Also don't forget to initialize the DC motors (refer to section C). As well as include and initialize the ultrasonic sensor (refer to section B). Continue by including the Software serial library, this library will allow us to appoint pins to act as RX (receiving pin) and TX (transmitting pin). The Software library can be used only after typing "#include <SoftwareSerial.h>" before the "Void setup()".

2. Assign pin 16 and 17 to be RX and TX pins respectively by typing the following code "SoftwareSerial Serial1(16,17);" before "void setup()". See the example below for reference.

```
#include <AFMotor.h>
#include <NewPing.h>

#define TRIGGER_PIN 6 // Ardunio pin tied to trigger pin on the ultreasonic sensor.
#define ECHO_PIN 5  //Ardunio pin tied to echo pin on the ultreasonic sensor.
#define MAX_DISTANCE 200 // Max distance to ping in cm
NewPing sonar (TRIGGER_PIN,ECHO_PIN,MAX_DISTANCE); // passing the pins to newping to creat sonar.
boolean prox;
int distance;
#include <SoftwareSerial.h>

AF_DCMotor rightMotor(1, MOTOR12_64KHZ);
AF_DCMotor leftMotor(2, MOTOR12_64KHZ);
SoftwareSerial Serial1(16,17);

void setup()
{
```

3. Now we need to initiate a rate for the bluetooth chip to communicate with the board inside the "void setup()". Ee will do that by typing "Serial1.begin(115200);" we chose 115200 because that is the default rate for both of the chips. You will also need to set the motor speed to 200 by typing NameOfYourMotor.setSpeed(200); See the example below for reference.

```
void setup()
{
  Serial1.begin(115200);
  rightMotor.setSpeed(200);
  leftMotor.setSpeed(200);
}

void loop()
{
```

4. Now inside the loop we need to continuously check for proximity to other object (refer to section b) as well as read the serial link to check whether we get the any directional commands. You will need to assign a char variable to store the letters received from the smartphone. See the example below for reference.

```
void loop()
{
  sensor_read(); // calling the function to determin if the robot is too close to an object.
if( prox == false);{
 if(Serial1.available()) // if theres data in the serial connection.
  {
    char c=Serial1.read(); // read and store data to c.
```

5. Add stamens to control the direction of the car by using the .run(FORWARD) or .run(BACKWARD) for each motor. See the example below for reference.

```
    {
      char c=Serial1.read(); // read and store data to c.

if(c=='f')// if c is f then Move front
{

rightMotor.run(FORWARD);
leftMotor.run(FORWARD);
}
if(c=='b')//if c is b then Move back
{
  rightMotor.run(BACKWARD);
  leftMotor.run(BACKWARD);
}
if(c=='l')//if c is l then Move left
{
  rightMotor.run(FORWARD);
  leftMotor.run(BACKWARD);
}
if(c=='r')//if c is r then Move Right
{
  leftMotor.run(FORWARD);
  rightMotor.run(BACKWARD);
}
if(c=='s')//if c is s then stop moving
{
  rightMotor.run(RELEASE);
  leftMotor.run(RELEASE);
}
}
}
}
void sensor_read(){
```

6. Add statements to control the sensor. Create a new function and call it “sensor_read()”. Have the function ping the sensor, print the distance to the serial port, and set the Boolean variable initialized earlier to “true” if the read distance is greater than 10. Remember that a read value of “0” means the distance is out of range. If there is something less than 10

cm away from the sensor, set the Boolean variable to “false”. See the example below for reference.

```
void sensor_read(){
  delay(250);
  distance= sonar.ping_cm();
  Serial1.println(distance);
  if(distance >10 or distance==0){
  Serial1.println("No Obstruction");
  prox= true;
  }else{Serial1.println(" Obstruction");
  prox= false;
}
}
```

7. Finally upload the sketch to the Arduino, and test by using the App which will be developed in a later lab. Be sure to use the serial monitor to read the sensor distance and the commands. Show the TA your work.

App Development

- Iphone **(option 1)**
- **OR** for older Android devices with any other version of Bluetooth (not LE) see **option 2** further down

Enable developer options (Android only)

1. Tap the Home button to go to your phone's Home screen.
2. Tap the Menu button, then Settings, then Applications.
3. If your phone has an Unknown sources setting, make sure it is checked.
4. Tap Development.
5. Make sure both USB Debugging and Stay Awake are checked.

The steps are similar for newer devices, but the options can be found in different places on the device: On Android 4.0* and newer, these settings are in Settings > Developer options.

Note: On Android 4.2* and newer, Developer options is hidden by default. To make it available, go to Settings > About phone and tap Build number seven times. Return to the previous screen to find Developer options.

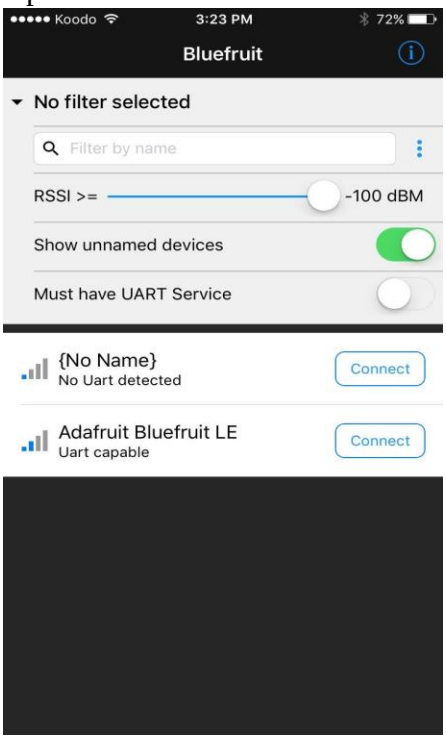
*****Unknown Soucres and USB Debugging may be in different directories in Settings depending on the Android version. Use the search button in Settings if you cannot locate them.

Option 1:

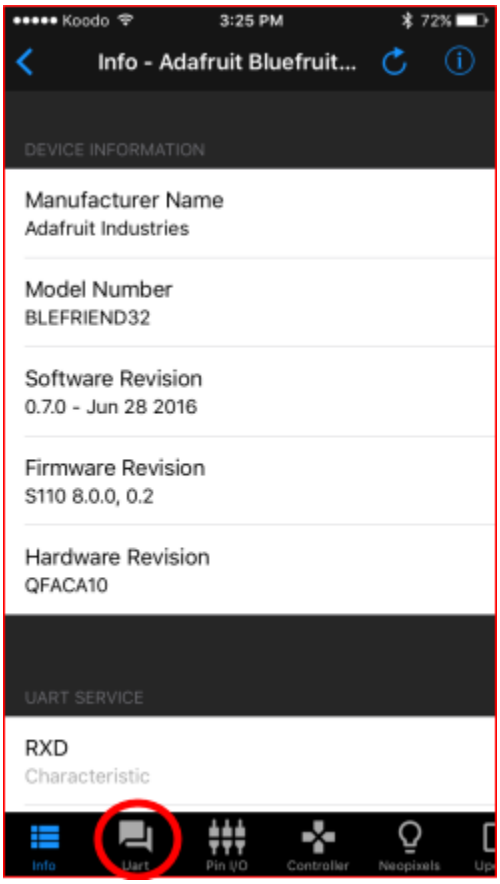
For IoS:

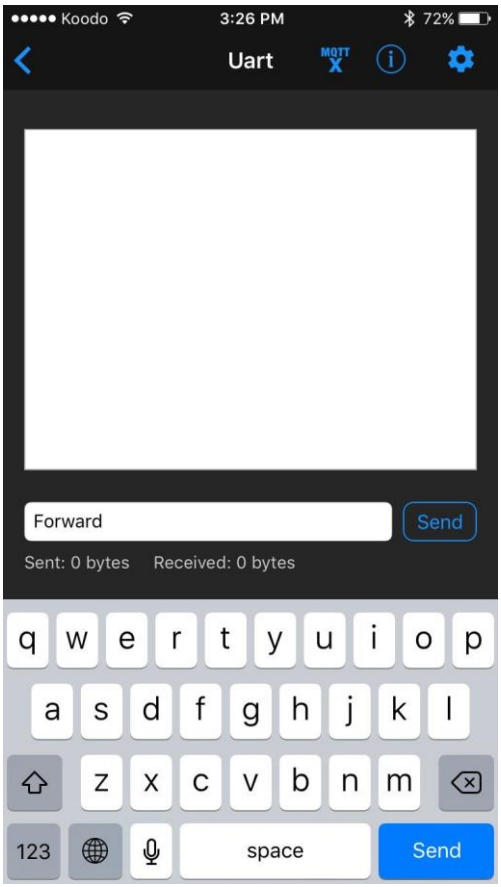
Make sure BLUETOOTH is on.

1. Go to the App store and install Bluefruit app.
2. Open it and select Adafruit Bluefruit LE device and Connect to it.



3. Then select the UART option at the bottom.





4. Type your commands (“forward”, “stop”, ...etc) in the box and click Send to command your car.
5. Show the TA your work.

Option 2:

For any other version of Bluetooth that is not Low Energy.

1. Open MIT App Inventor by going to http://appinventor.mit.edu/explore/ and clicking on the “Create Apps!” button. Select a Gmail account to use (@uottawa.ca accounts work). Once logged in, create and name a new project.
2. Now start creating the user interface. To add an object to the app, drag the corresponding icon from the left hand menu into the prototype build screen. The following objects are required for this introductory app.

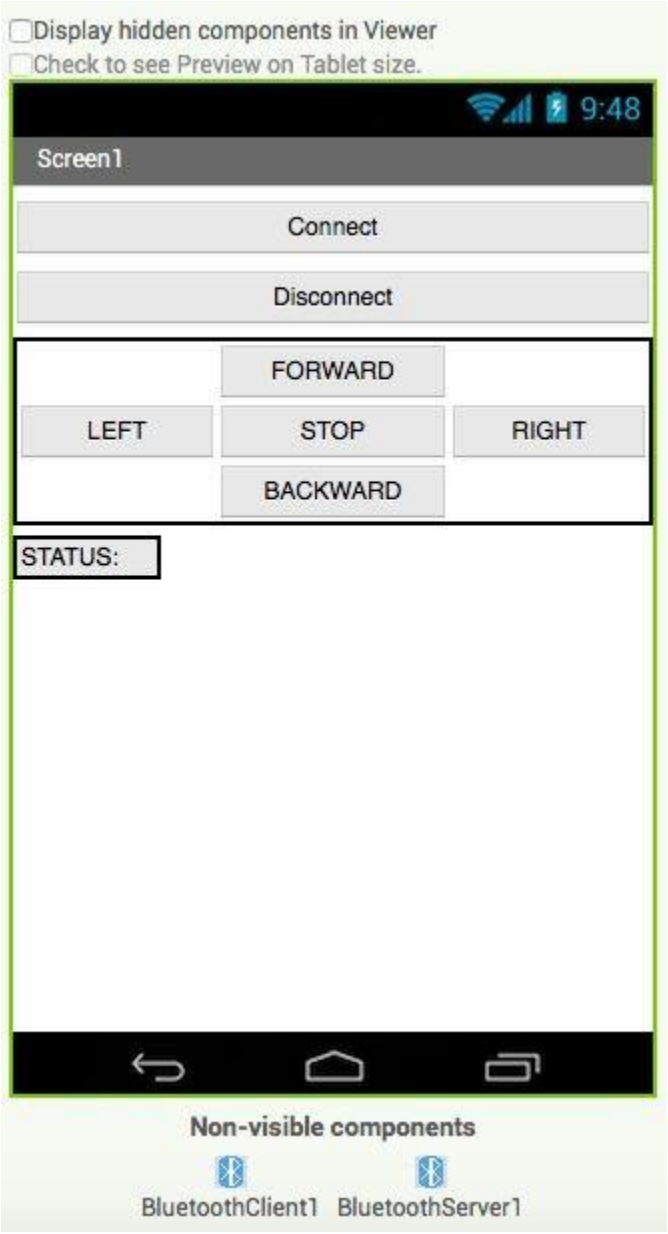
   For Bluetooth connectivity:

   - One button, call it “Disconnect”
   - A list picker object
   - A regular Bluetooth client and Bluetooth server

For control:

- 4 buttons for directions
- A label
- A stop button

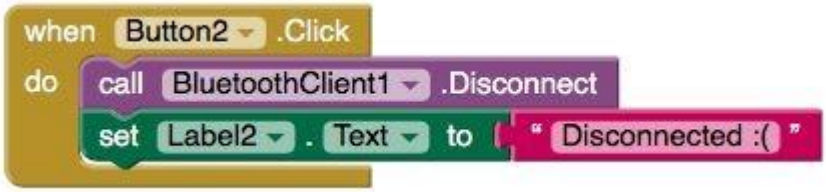
3. Use the layout functions to arrange the input objects. Use the table arrangement for the directional control (hint: use a 3x3 table), and the horizontal arrangement for connectivity buttons.
    - Note: change the size of the tables (fill parent) before adding the buttons
4. Change the default text of the input object to reflect their intended function. This can be done in the right hand window, once an object (such as a button or label) is selected. Below is an example of a user interface used in this app. Show the TA your completed user interface.



5. Program the app's functionality. Switch to the "Blocks" tab using the button in the right hand corner. This shows the coding environment used by the app inventor.
6. First Program the listpicker element. Choose the "When ListPicker1.BeforePicking" option under the ListPicker1 object, and nest "set ListPicker1.Elements to" from the list picker object's menu. Connect the "BluetoothClient1.AddressesAndNames" block (found in the BluetoothClient menu) to the end of the list picker block, as seen below. This will display all found devices in a drop down list (the list picker object).

7. Once a device has been selected from the list by the user, it must be connected to by the smartphone. From the list picker command menu, select “when ListPicker1.AfterPicking”. This block dictates what happens after a device has been chosen from the list.
8. Create a “set ListPicker1.Seclection to” block from the ListPicker1 menu on the left. From the “BluetoothClient1” menu, choose the “call BluetoothClient1.Connect address” block and connect to it the “ListPicker1.Selection” block from the list picker menu. Select “set Label2.Text to” from the label menu and then select text from the left menu, write “Connected :)” and add it to label. (The picture below may be used for reference)

9. Now select a “when Button.click do” block from one of the button menu’s, and attach a “call BluetoothClient1.Disconnect” block to it. This will disconnect the smartphone. Now select “set Label2.Text to” from the label menu and then select text from the left menu, write “Disconnected :(” and add it to label. (The picture below may be used for reference)



10. Moving on to the control Buttons. Select a “when Button?.click do” (? being the number of the button) and nest in it the “call BluetoothClient1.SendText” from the bluetooth menu. Add a text object with the direction you would like it to move upon pressing that button. (The picture below may be used for reference)

11. To test the app on an android device, download and save the app as an .apk file (select “Build” at the top of the webpage). Email the file to an email address available on the phone and open the file. You may have to allow the phone to install third-party applications. The app should then install, and can be run as a normal app when finished. You may also prompt MIT App Inventor to display a QR code. Scan the code with the

android phone (MIT AI2 Companion), and the app should download automatically. Show the TA your work.