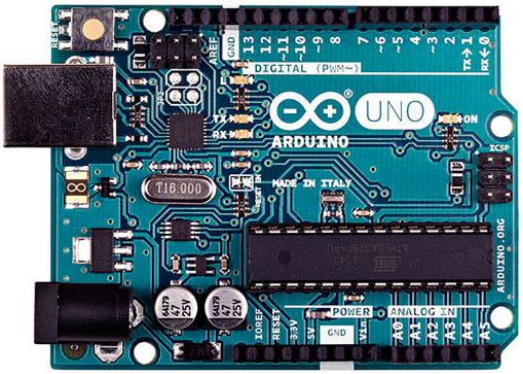
HARDWARE DESIGN

i) ARDUINO UNO

Basically it is an microcontroller based on architecture of ATmega328P that are being program with Processing language that has been modified that later being called Arduino IDE.



The Arduino UNO are remarkable for enthusiast in electronic as it is affordable to obtain while still providing the necessary features such as PWM, ADC, Digital I/O, I^2C, SPI and Serial Connectivity.

The device are the key player in this project as it would work on the process required from the multiple input and output from sensors, LCD and Android phone.

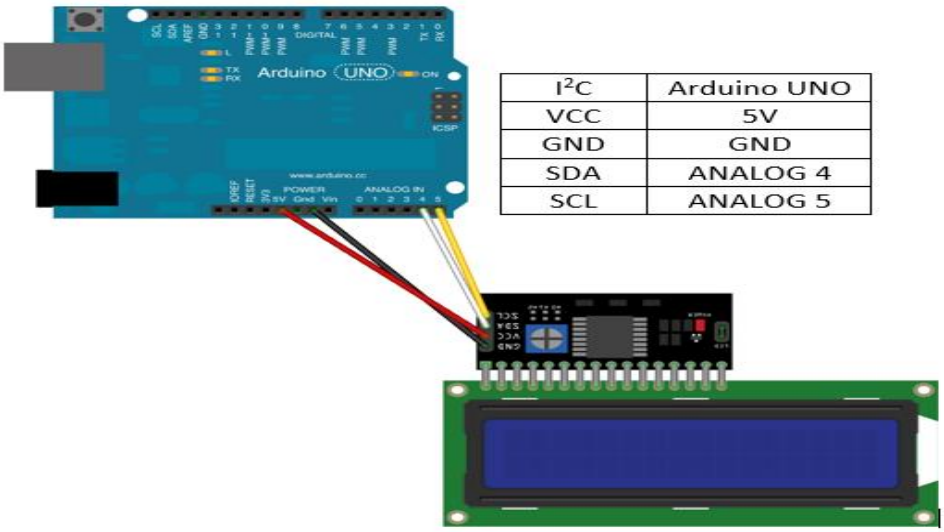
ii) LCD DISPLAY / I^2C

LCD stand for Liquid Crystal Display, is a component with a flat panel, electronic visual display that uses light modulating properties of liquid crystal. LCDs are available to display arbitrary images as in a general-purpose computer display or fixed images with low information content, which can be displayed or hidden, such as pre-set words, characters, digits or 7 segment displays.

The function of using LCD in our designed Autonomous System is to show the current value of percentage that we tuned based on analogue value from potentiometer (act as controller/sensor).

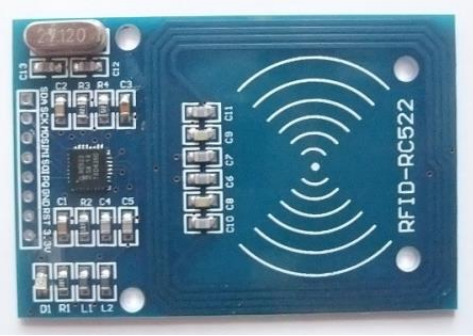
Inter-Intergrated Circuit (I^2C) is typically used for attaching lower-speed peripheral ICs to processors and microcontroller in short distance, intra-board communication. I^2C was connected to LCD to reduce the number of pins used before connected to Arduino. With this, only 4 pins connected to Arduino which are Vcc, Gnd, SDA and SCL.

To command or write the program code for LCD with I^2C, it need a LiquidCrystal_I2C.h as it library header along with initializing the LCD with LiquidCrystal_I2C lcd(0x27, 16, 2); After that, just use the command for LCD as usual.



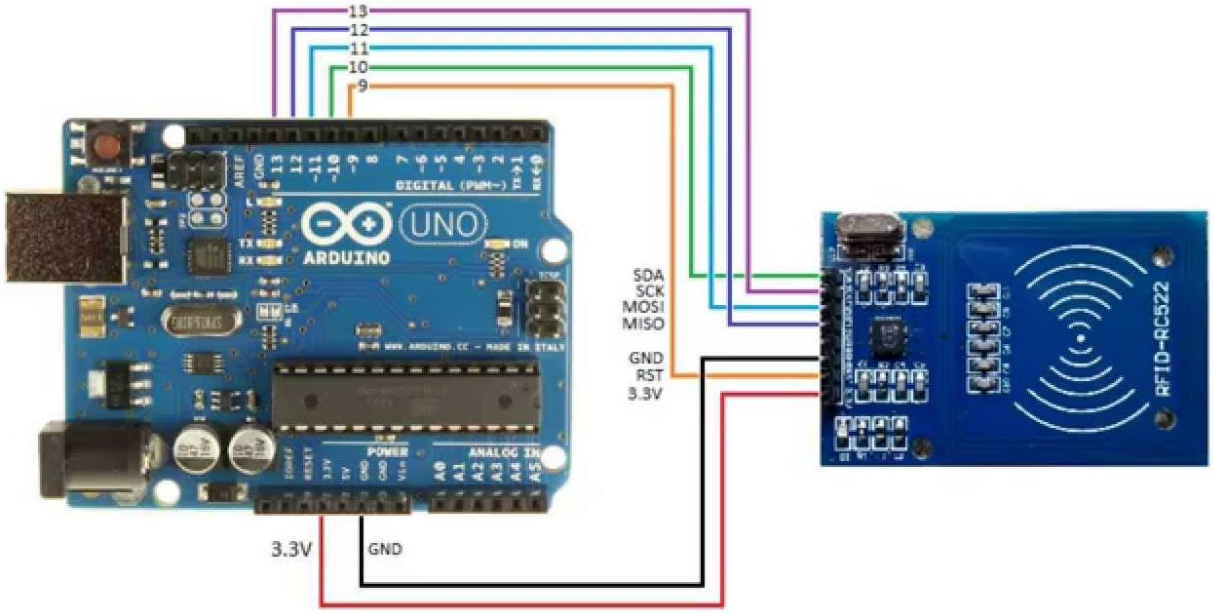
I²C	Arduino UNO
VCC	5V
GND	GND
SDA	ANALOG 4
SCL	ANALOG 5

iii) RFID MODULE

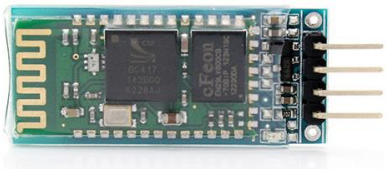


Radio frequency identification which known as RFID, is the use of radio waves to read and capture information stored on a tag that attached to an object. RFID modules use electromagnetic fields to transfer data between card and the reader.

Our project use read-write data RC522 model, based on NPX chip MFRC522. The specification tag is Mifare MF15503 with 1 kilobyte EEPROM. It works on Serial Peripheral Interface (ISP) protocol, when interfaced with Arduino board with operating frequency, 13.56 MHz.

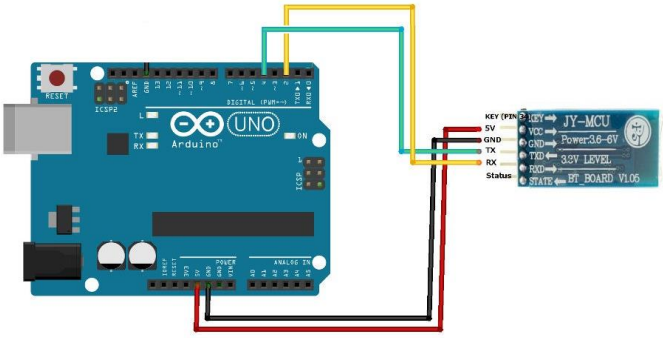


iv) HC06 BLUETOOTH MODULE



Bluetooth are vastly being known as a medium for communication from one end to another and its own credibility on transmitting the data on a low wattage.

It is a wireless technology that used UHF radio waves (2.4 GHz – 2.485 GHz) that would travel in a short distance. The range are usually around 5 m to 400 m depend on the configuration and architecture on board.



In this project, Bluetooth module are crucial element as it are needed to communicate between Arduino UNO and Android phone through an built application. The Bluetooth are connected serially to the Arduino using pin TX and RX.

SOFTWARE DESIGN

i) ANDROID STUDIO SOFTWARE

It is an official IDE for the android application development tool after Eclipse Android Development Tools (ADT) with tons of enhance features and improvement that benefit to developer itself and the end user. The software are based on IntelliJ IDEA with Java language as the core for the application processing. The layout or the interface of the user interface (UI) are being written in XML file while it is compatible to be combined with other UI written language such as HTML, CSS and etc.

As the rubric mentioned the extra marks on the software used/created, our team decided to develop an application that will enable the two way communication between Arduino and Android. Since the project are about automated agriculture, the application are named as AgriCU that stand for ‘Agriculture Control Unit’.

The develop application are expected to be able to give several specific command to Arduino through Bluetooth communication such as light intensity control, water dispenser control, enclosed temperature control and data display command while continuously receiving data on the Arduino sensor and display it on Android phone when called.

Arduino UNO Code

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "AddicoreRFID.h"
#include <SPI.h>

#define  uchar unsigned char
#define uint  unsigned int

uchar fifobytes;
uchar fifoValue;

AddicoreRFID myRFID;

const int chipSelectPin = 10;
const int NRSTPD = 5;

//Maximum length of the array
#define MAX_LEN 16

LiquidCrystal_I2C lcd(0x27,16,2);

int led = 3;
int tim = 10;
int value_roof;
int percent_roof;
int value_water;
int percent_water;
int value_humidity;
int percent_humidity;
int a=0, b=0, receivedata=0;


const int analogInPin0 = A0;
const int analogInPin1 = A1;
const int analogInPin2 = A2;

float sensorValue[3] = {0,0,0};
float voltageValue[3] = {0,0,0};

char inbyte = 0,btnbyte;

int roofSensor, waterSensor;

void setup(){
  Serial.begin(9600);
  initLCD();
  initRFID();

  pinMode(led, OUTPUT);
  digitalWrite(led, LOW);

}

void loop() {
  readSensors();
  getVoltageValue();
  sendAndroidValues();
  workLCD();
  workRFID();
  btnbyte = Serial.read();
}

//:::::::::::::::::::::::::::RFID-COMMAND::::::::::::::::::::::::::::::::

void withSecurity(void){
  if (btnbyte == '0'){
    digitalWrite(led, LOW);
    }
  if (btnbyte == '1'){
    digitalWrite(led, HIGH);
  }
}


void withoutSecurity(void){
  if (btnbyte == '0'){
    digitalWrite(led, HIGH);
  }
  if (btnbyte == '1'){
    digitalWrite(led, LOW);
  }
}
//^^^^^^:::::::::::RFID-COMMAND::::::::::::::^^^^^^

//:::::::::::::::::::::::::::SENSOR::::::::::::::::::::::::::::::::

void readSensors(){
  // read the analog in value to the sensor array
  sensorValue[0] = analogRead(analogInPin0);
  sensorValue[1] = analogRead(analogInPin1);
  sensorValue[2] = analogRead(analogInPin2);
}

void getVoltageValue(){
  for (int x = 0; x < 3; x++){
    voltageValue[x] = ((sensorValue[x]/1023)*100);
  }
}

//sends the values from the sensor over serial to BT module
void sendAndroidValues(){
      //puts # before the values so our app knows what to do with
the data
    Serial.print('#');
    //for loop cycles through 4 sensors and sends values via serial
    for(int k=0; k<3; k++){
      Serial.print(voltageValue[k]);
      Serial.print('+');
      //technically not needed but I prefer to break up data values
      //so they are easier to see when debugging
    }
    Serial.print('~'); //used as an end of transmission character -
used in app for string length
    Serial.println();
    delay(10);        //added a delay to eliminate missed
transmissions
}
//::::::::::^^^^^^:::::::::::SENSOR::::::::::::::^^^^^^::::::::::::::

//:::::::::::::::::::::::::::LCDD::::::::::::::::::::::::::::::::

void initLCD(void) {
  lcd.init(); //initialize the lcd
  lcd.backlight(); //open the backlight

  lcd.setCursor(0,0); // set the cursor to column 15, line 0
  lcd.print("Roof : "); // Print a message to the LCD.
  lcd.setCursor(0,1); // set the cursor to column 15, line 0
  lcd.print("Water: "); // Print a message to the LCD.
}


void workLCD(void) {
  value_roof=sensorValue[0];
  percent_roof=((value_roof/1023.0)*100);
  value_water=sensorValue[1];
  percent_water=((value_water/1023.0)*100);

  lcd.setCursor(7,0); // set the cursor to column 15, line 0
  lcd.print(percent_roof);// Print value of percent_roof
  lcd.print("%");

  lcd.setCursor(7,1); // set the cursor to column 15, line 0
  lcd.print(percent_water);// Print value of percent_water
  lcd.print("%");

  delay(tim); //wait for 250 microseconds

   if(percent_roof==30)//data roof dari fon
   {
    if(a == 0){
```

```
      lcd.setCursor(12,0);
      lcd.print("Done");
      a = 1;
    }
  }
  if(percent_water==60)//data water dari fon
  {
    if(b == 0){
      lcd.setCursor(12,1);
      lcd.print("Done");
      b = 1;
    }
  }
  else{
    lcd.setCursor(12,0);
    lcd.print("    ");
    lcd.setCursor(12,1);
    lcd.print("    ");
    a = 0;
    b = 0;
  }
}
//::::::::::^^^^^^::::::::::LCDD::::::::::::^^^^^^::::::::::::::

//:::::::::::::::::::::::::::RFID::::::::::::::::::::::::::::::::

void initRFID(void){
  // start the SPI library:
  SPI.begin();

  pinMode(chipSelectPin,OUTPUT);              // Set digital pin 10
as OUTPUT to connect it to the RFID /ENABLE pin
  digitalWrite(chipSelectPin, LOW);         // Activate the RFID
reader
  pinMode(NRSTPD,OUTPUT);                     // Set digital pin
10 , Not Reset and Power-down
  digitalWrite(NRSTPD, HIGH);

  myRFID.AddicoreRFID_Init();
}


void workRFID(void){
  uchar i, tmp, checksum1;
  uchar status;
  uchar str[MAX_LEN];
  uchar RC_size;
  uchar blockAddr;  //Selection operation block address 0 to 63
  String mynum = "";

  str[1] = 0x4400;

  //Find tags, return tag type
  status = myRFID.AddicoreRFID_Request(PICC_REQIDL,
str);

  //Anti-collision, return tag serial number 4 bytes
  status = myRFID.AddicoreRFID_Anticoll(str);
  if (status == MI_OK)
  {
    if(str[0] == 64)                    //You can change this to the first
byte of your tag by finding the card's ID through the Serial
Monitor
    {
      withSecurity();
    }
  }
  else if(status == MI_NO_TAG_ERR){
    withoutSecurity();
  }
  myRFID.AddicoreRFID_Halt();      //Command tag into
hibernation
}
//::::::::::^^^^^^::::::::::RFID::::::::::::^^^^^^::::::::::::::
```

Android Studio Code

JAVA File:
Device List Activity

```
package com.airul.agricu_v2;

import java.util.Set;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;

public class DeviceListActivity extends Activity {
    // Debugging for LOGCAT
    private static final String TAG = "DeviceListActivity";
    private static final boolean D = true;

    // declare button for launching website and textview for
connection status
    Button tlbutton;
    TextView textView1;

    // EXTRA string to send on to mainactivity
    public static String EXTRA_DEVICE_ADDRESS =
"device_address";

    // Member fields
    private BluetoothAdapter mBtAdapter;
    private ArrayAdapter<String> mPairedDevicesArrayAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.device_list);
    }

    @Override
    public void onResume()
    {
        super.onResume();
        //**********
        checkBTState();

        textView1 = (TextView) findViewById(R.id.connecting);
        textView1.setTextSize(40);
        textView1.setText(" ");

        // Initialize array adapter for paired devices
        mPairedDevicesArrayAdapter = new
ArrayAdapter<String>(this, R.layout.device_name);

        // Find and set up the ListView for paired devices
        ListView pairedListView = (ListView)
findViewById(R.id.paired_devices);
        pairedListView.setAdapter(mPairedDevicesArrayAdapter);

pairedListView.setOnItemClickListener(mDeviceClickListener)
;
        // Get the local Bluetooth adapter
        mBtAdapter = BluetoothAdapter.getDefaultAdapter();
        // Get a set of currently paired devices and append to
'pairedDevices'
        Set<BluetoothDevice> pairedDevices =
mBtAdapter.getBondedDevices();
        // Add previosuly paired devices to the array
        if (pairedDevices.size() > 0) {
findViewById(R.id.title_paired_devices).setVisibility(View.VIS
IBLE);//make title viewable
            for (BluetoothDevice device : pairedDevices) {
                mPairedDevicesArrayAdapter.add(device.getName()
+ "\n" + device.getAddress());
            }
        } else {
            String noDevices =
getResources().getText(R.string.none_paired).toString();
            mPairedDevicesArrayAdapter.add(noDevices);
        }
    }
    // Set up on-click listener for the list (nicked this - unsure)
    private OnItemClickListener mDeviceClickListener = new
OnItemClickListener() {
        public void onItemClick(AdapterView<?> av, View v, int
arg2, long arg3) {
            textView1.setText("Connecting...");
            // Get the device MAC address, which is the last 17 chars
in the View
            String info = ((TextView) v).getText().toString();
            String address = info.substring(info.length() - 17);
            // Make an intent to start next activity while taking an
extra which is the MAC address.
            Intent i = new Intent(DeviceListActivity.this,
MainActivity.class);
            i.putExtra(EXTRA_DEVICE_ADDRESS, address);
            startActivity(i);
        }
    };
    private void checkBTState() {
        // Check device has Bluetooth and that it is turned on
        mBtAdapter=BluetoothAdapter.getDefaultAdapter(); //
CHECK THIS OUT THAT IT WORKS!!!
        if(mBtAdapter==null) {
            Toast.makeText(getBaseContext(), "Device does not
support Bluetooth", Toast.LENGTH_SHORT).show();
        } else {
            if (mBtAdapter.isEnabled()) {
                Log.d(TAG, "...Bluetooth ON...");
            } else {
                //Prompt user to turn on Bluetooth
                Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(enableBtIntent, 1);
            }
        }
    }
}
```

Android Studio Code

JAVA File:
Main Activity

```
package com.airul.agricu_v2;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Timer;
import java.util.TimerTask;
import java.util.UUID;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;

public class MainActivity extends Activity {
   Button btnOn, btnOff, btnData, btnRoof, btnWater, btnTemp;
   TextView txtArduino, txtString, txtStringLength, textView;
   Handler bluetoothIn;
   final int handlerState = 0;                        //used to identify
handler message
   int point = 0, receivedata = 0;
   private BluetoothAdapter btAdapter = null;
   private BluetoothSocket btSocket = null;
   private StringBuilder recDataString = new StringBuilder();
   private ConnectedThread mConnectedThread;
   private OutputStream outputStream;
   // SPP UUID service - this should work for most devices
   private static final UUID BTMODULEUUID =
UUID.fromString("00001101-0000-1000-8000-
00805F9B34FB");
   // String for MAC address
   private static String address;
   private static SeekBar seek_bar;
   private static TextView text_view;
   private static ToggleButton toggle;
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      //Link the buttons and textViews to respective views
      btnOn = (Button) findViewById(R.id.buttonON);
      btnOff = (Button) findViewById(R.id.buttonOff);
      btnData = (Button) findViewById(R.id.buttonData);
      btnRoof = (Button) findViewById(R.id.buttonRoof);
      btnTemp = (Button) findViewById(R.id.buttonTemp);
      btnWater = (Button) findViewById(R.id.buttonWater);
      txtString = (TextView) findViewById(R.id.txtString);
      txtStringLength = (TextView)
findViewById(R.id.testView1);
      seek_bar = (SeekBar)findViewById(R.id.seekBar);
      text_view
=(TextView)findViewById(R.id.textViewSlider);
      ToggleButton toggle = (ToggleButton)
findViewById(R.id.toggleButton);
      seekBar();
      textView = (TextView) findViewById(R.id.textView);
      bluetoothIn = new Handler() {
         public void handleMessage(android.os.Message msg) {
            if (msg.what == handlerState) {
//if message is what we want
               String readMessage = (String) msg.obj;
// msg.arg1 = bytes from connect thread
               recDataString.append(readMessage);
//keep appending to string until ~

               //:::::::::::::::::START::::::::::::::::
int endOfLineIndex = recDataString.indexOf("~");                    //
determine the end-of-line
if (endOfLineIndex > 0) {                                           // make
sure there data before ~
String dataInPrint = recDataString.substring(0,
endOfLineIndex);    // extract string
if (recDataString.charAt(0) == '#')                                //if it starts
with # we know it is what we are looking for
{
   String sensor0 = recDataString.substring(1, 6);             //get
sensor value from string between indices 1-5
   String sensor1 = recDataString.substring(7, 12);            //same
again...
   String sensor2 = recDataString.substring(13, 18);
   textView.setTextSize(30);
   textView.setText("Light Intensity\t\t= " + sensor0 + "%\n" +
                           "Temperature\t\t\t= " + sensor1 +
"%\n" +
                           "Humidity\t\t\t\t\t\t\t= " + sensor2 +
"%\n");
                  }
else if (recDataString.charAt(0) == '$') {
                  }recDataString.delete(0,
recDataString.length());
                }
            }
         }
      };

      btAdapter = BluetoothAdapter.getDefaultAdapter();       //
get Bluetooth adapter
      checkBTState();

      toggle.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
         @Override
         public void onCheckedChanged(CompoundButton
buttonView, boolean isChecked) {
            if (isChecked) {
               // The toggle is enabled
               mConnectedThread.write("1");    // Send "1" via
Bluetooth
               Toast.makeText(getBaseContext(), "Turn on LED",
Toast.LENGTH_SHORT).show();

            } else {
               // The toggle is disabled
               mConnectedThread.write("0");    // Send "0" via
Bluetooth
               Toast.makeText(getBaseContext(), "Turn off LED",
Toast.LENGTH_SHORT).show();

            }
         }
      });

      // Set up onClick listeners for buttons to send 1 or 0 to turn
on/off LED
      btnOff.setOnClickListener(new OnClickListener() {
         public void onClick(View v) {
            mConnectedThread.write("0");    // Send "0" via
Bluetooth
            Toast.makeText(getBaseContext(), "Turn off LED",
Toast.LENGTH_SHORT).show();
         }
      });

      btnOn.setOnClickListener(new OnClickListener() {
         public void onClick(View v) {
```

```
            mConnectedThread.write("1");    // Send "1" via
Bluetooth
            Toast.makeText(getBaseContext(), "Turn on LED",
Toast.LENGTH_SHORT).show();
          }
       });
   }

   private BluetoothSocket
createBluetoothSocket(BluetoothDevice device) throws
IOException {

      return
device.createRfcommSocketToServiceRecord(BTMODULEUU
ID);
      //creates secure outgoing connecetion with BT device using
UUID
   }


   @Override
   public void onResume() {
      super.onResume();

      //Get MAC address from DeviceListActivity via intent
      Intent intent = getIntent();

      //Get the MAC address from the DeviceListActivty via
EXTRA
      address =
intent.getStringExtra(DeviceListActivity.EXTRA_DEVICE_A
DDRESS);
      //create device and set the MAC address
      BluetoothDevice device =
btAdapter.getRemoteDevice(address);

      try {
         btSocket = createBluetoothSocket(device);
      } catch (IOException e) {
         Toast.makeText(getBaseContext(), "Socket creation
failed", Toast.LENGTH_LONG).show();
      }
      // Establish the Bluetooth socket connection.
      try
      {
         btSocket.connect();
      } catch (IOException e) {
         try
         {
            btSocket.close();
         } catch (IOException e2)
         {
            //insert code to deal with this
         }
      }
      mConnectedThread = new ConnectedThread(btSocket);
      mConnectedThread.start();

      //I send a character when resuming.beginning transmission
to check device is connected
      //If it is not an exception will be thrown in the write method
and finish() will be called
      mConnectedThread.write("x");
   }

   @Override
   public void onPause() {
      super.onPause();
      try
      {
         //Don't leave Bluetooth sockets open when leaving
activity
         btSocket.close();
      } catch (IOException e2) {
         //insert code to deal with this
      }
   }

   //Checks that the Android device Bluetooth is available and
prompts to be turned on if off
   private void checkBTState() {

      if(btAdapter==null) {
         Toast.makeText(getBaseContext(), "Device does not
support bluetooth", Toast.LENGTH_LONG).show();
      } else {
         if (btAdapter.isEnabled()) {
         } else {
            Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBtIntent, 1);
         }
      }
   }

   //create new class for connect thread
   private class ConnectedThread extends Thread {
      private final BluetoothSocket mmSocket;        //NEW
      private final InputStream mmInStream;
      private final OutputStream mmOutStream;

      //creation of the connect thread
      public ConnectedThread(BluetoothSocket socket) {
         mmSocket = socket;                          //NEW
         InputStream tmpIn = null;
         OutputStream tmpOut = null;

         try {
            //Create I/O streams for connection
            tmpIn = socket.getInputStream();
            tmpOut = socket.getOutputStream();
         } catch (IOException e) { }

         mmInStream = tmpIn;
         mmOutStream = tmpOut;
      }

      public void run() {
         byte[] buffer = new byte[1024];             //NEW
         int bytes;

         // Keep looping to listen for received messages
         while (true) {
            try {
               bytes = mmInStream.read(buffer);          //read
bytes from input buffer
               String readMessage = new String(buffer, 0, bytes);
               // Send the obtained bytes to the UI Activity via
handler
               bluetoothIn.obtainMessage(handlerState, bytes, -1,
readMessage).sendToTarget();
            } catch (IOException e) {
               break;
            }
         }
      }
      //write method
      public void write(String input) {
         byte[] msgBuffer = input.getBytes();          //converts
entered String into bytes
         try {
            mmOutStream.write(msgBuffer);              //write
bytes over BT connection via outstream
         } catch (IOException e) {
            //if you cannot write, close the application
            Toast.makeText(getBaseContext(), "Connection
Failure", Toast.LENGTH_LONG).show();
            finish();

         }
```

```
    }

  }

  public void seekBar( ){

    text_view.setText(seek_bar.getProgress() + " %");

    seek_bar.setOnSeekBarChangeListener(
        new SeekBar.OnSeekBarChangeListener() {

          int progress_value;
          @Override
          public void onProgressChanged(SeekBar seekBar,
int progress, boolean fromUser) {
            progress_value = progress;
            text_view.setText(seek_bar.getProgress() + "
%");
//            Toast.makeText(MainActivity.this,"SeekBar in
progress",Toast.LENGTH_LONG).show();
          }

          @Override
          public void onStartTrackingTouch(SeekBar
seekBar) {
//            Toast.makeText(MainActivity.this,"SeekBar in
StartTracking",Toast.LENGTH_LONG).show();
          }

          @Override
          public void onStopTrackingTouch(SeekBar
seekBar) {
            text_view.setText(seek_bar.getProgress() + "
%");
//            Toast.makeText(MainActivity.this,"SeekBar in
StopTracking",Toast.LENGTH_LONG).show();
          }
        }
    );

  }
}
```