INTRODUCTION

INTRODUCTION

The IR Remote Emulator project that can help all variety of people from children to elders whoever uses an IR based remote.This system allows the user to add any of the IR compatible remote to the provided application by using configuration process and then use them in a easy and simple way with the help of mobile phone.

The system has an User Interface which makes the project so specific for its functions and has a wonderful interface to meet the user's needs and this also takes the data automatically if the option IR Receiver is enabled.

This system has two important sensors namely IR Transmitter and an IR Receiver.

The IR Receiver is used to get and decode IR signals from the remote which is to be used with the mobile phone and IR Transmitter is used to regenerate the available codes when needed. The system uses an application named "Arduino IR Kit" which is used for most of the running operations in our project.

SYSTEM SPECIFICATION

SYSTEM SPECIFICATION

The Hardware Configuration involved in this project is

PC

Processor : Intel Core i3 Processor or Higher

RAM : 4GB

Hard Disk : 500GB

Monitor : 17" Color Monitor

MOBILE

Processor : Snapdragon 425 or Higher

RAM : 2 GB

Android : Android 4.0 and Higher

HARDWARE SPECIFICATION

Arduino Uno : 1 No

General Purpose Transistor NPN : 1 No

Resistor 1k ohm : 1 No.

Bluetooth Low Energy Module : 1 No.

Infrared Emitter : 1 No.

Infrared Receiver 38 kHz : 1 No

SOFTWARE SPECIFICATION

The software requirements to develop the project are

Operating System : Windows 7 and higher

Front End : Arduino IDE

Applications : Arduino IR Kit(Android)
Arduino IR Kit(iOS)

PROJECT DESCRIPTION AND DESIGN

FRONT END – ARDUNIO IDE

The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as **Windows, Mac OS X, and Linux**. It supports the programming languages C and C++. Here, IDE stands for **Integrated Development Environment**.

The program or code written in the Arduino IDE is often called as sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'

Let's discuss each section of the Arduino IDE display in detail.

Toolbar Button

The icons displayed on the toolbar are **New, Open, Save, Upload,** and **Verify**.

Upload

The Upload button compiles and runs our code written on the screen. It further uploads the code to the connected board. Before uploading the sketch, we need to make sure that the correct board and ports are selected.

We also need a USB connection to connect the board and the computer. Once all the above measures are done, click on the Upload button present on the toolbar.

The latest Arduino boards can be reset automatically before beginning with Upload. In the older boards, we need to press the Reset button present on it. As soon as the uploading is done successfully, we can notice the blink of the Tx and Rx LED.

If the uploading is failed, it will display the message in the error window.

We do not require any additional hardware to upload our sketch using the Arduino Bootloader. A **Bootloader** is defined as a small program, which is loaded in the microcontroller present on the board. The LED will blink on PIN 13.

- **Open**

The Open button is used to open the already created file. The selected file will be opened in the current window.

- **Save**

The save button is used to save the current sketch or code.

- **New**

It is used to create a new sketch or opens a new window.

- **Verify**

The Verify button is used to check the compilation error of the sketch or the written code.

Serial Monitor

The serial monitor button is present on the right corner of the toolbar. It opens the serial monitor.

When we connect the serial monitor, the board will reset on the operating system Windows, Linux, and Mac OS X. If we want to process the control characters in our sketch, we need to use an external terminal program. The terminal program should be connected to the COM port, which will be assigned when we connect the board to the computer.

Menu Bar

- **File**

When we click on the File button on the Menu bar, a drop-down list will appear

- **New**

The New button opens the new window. It does not remove the sketch which is already present.

- **Open**

It allows opening the sketch, which can be browsed from the folders and computer drivers.

- **Open Recent**

The Open Recent button contains the list of the recent sketches.

- **Sketchbook**

It stores the current sketches created in the Arduino IDE software. It opens the selected sketch or code in a new editor at an instance.

Examples

It shows the different examples of small projects for a better understanding of the IDE and the board. The IDE provides examples of self-practice.

- **Close**

The Close button closes the window from which the button is clicked.

- **Save**

The save button is used to save the current sketch. It also saves the changes made to the current sketch. If we have not specified the name of the file, it will open the '**Save As...**' window.

- **Save As...**

We can save the sketch with a different name using the '**Save As...**' button. We can also change the name accordingly.

- **Page Setup**

It allows setting the page margins, orientation, and size for printing. The '**Page Setup**'

- **Print**

According to the settings specified in the 'Page Setup', it prepares the current sketch for printing.

Preferences

It allows the customization settings of the Arduino IDE.

Quit

The Quit button is used to close all the IDE windows. The same closed sketch will be reopened when we will open the Arduino IDE.

- **Edit**

When we click on the Edit button on the Menu bar, a drop-down list appears

Let's discuss each option in detail.

- **Undo**

The Undo button is used to reverse the last modification done to the sketch while editing.

- **Redo**

The Redo button is used to repeat the last modification done to the sketch while editing.

- **Cut**

It allows us to remove the selected text from the written code. The text is further placed to the clipboard. We can also paste that text anywhere in our sketch.

- **Copy**

It creates a duplicate copy of the selected text. The text is further placed on the clipboard.

- **Copy for Forum**

The 'Copy for Forum' button is used to copy the selected text to the clipboard, which is also suitable for posting to the forum.

- **Copy as HTML**

The 'Copy for Forum' button is used to copy the selected text as HTML to the clipboard. It is desirable for embedding in web pages.

- **Paste**

The Paste button is used to paste the selected text of the clipboard to the specified position of the cursor.

Select All

It selects all the text of the sketch.

Go to line...

It moves the cursor to the specified line number.

Comment/Decomment

The Comment/ Decomment button is used to put or remove the comment mark (//) at the beginning of the specified line.

Increase Indent

It is used to add the space at the starting of the specified line. The spacing moves the text towards the right.

Decrease Indent

It is used to subtract or remove the space at the starting of the specified line. The spacing moves the text towards the left.

Increase Font Size

It increases the font size of the written text.

Decrease Font Size

It decreases the font size of the written text.

Find...

It is used to find the specified text. We can also replace the text. It highlights the text in the sketch.

Find Next

It highlights the next word, which has specified in the '**Find...**' window. If there is no such word, it will not show any highlighted text.

Find Previous

It highlights the previous word, which has specified in the '**Find...**' window. If there is no such word, it will not show any highlighted text.

- **Sketch**

When we click on the Sketch button on the Menu bar, a drop-down list appears

Let's discuss each option in detail.

Verify/Compile

It will check for the errors in the code while compiling. The memory in the console area is also reported by the IDE.

Upload

The Upload button is used to configure the code to the specified board through the port.

Upload Using Programmer

It is used to override the Bootloader that is present on the board. We can utilize the full capacity of the Flash memory using the '**Upload Using Programmer**' option. To implement this, we need to restore the Bootloader using the **Tools-> Burn Bootloader** option to upload it to the USB serial port.

Export compiled Binary

It allows saving a .**hex** file and can be kept archived. Using other tools, .hex file can also be sent to the board.

Show Sketch Folder

It opens the folder of the current code written or sketch.

Include Library

Include Library includes various Arduino libraries. The libraries are inserted into our code at the beginning of the code starting with the #. We can also import the libraries from .zip file.

Add File...

The Add File... button is used to add the created file in a new tab on the existing file.

For example, let's add '**Blink**' file to the '**Javatpoint**' file.

We can also delete the corresponding file from the tab by clicking on the **small triangle** -> **Delete** option.

Tools

When we click on the Tools button on the Menu bar, a drop-down list appears

Let's discuss each option in detail.

Auto Format

The Auto Format button is used to format the written code. For example, lining the open and closed curly brackets in the code.

Archive Sketch

The copy of the current sketch or code is archived in the .zip format. The directory of the archived is same as the sketch.

Fix Encoding and Reload

This button is used to fix the inconsistency between the operating system char maps and editor char map encoding.

Manage Libraries...

It shows the updated list of all the installed libraries. We can also use this option to install a new library into the Arduino IDE.

Serial Monitor

It allows the exchange of data with the connected board on the port.

Serial Plotter

The Serial Plotter button is used to display the serial data in a plot. It comes preinstalled in the Arduino IDE.

WiFi101/WiFiNINA Firmware Updater

It is used to check and update the Wi-Fi Firmware of the connected board.

Board

We are required to select the board from the list of boards. The selected board must be similar to the board connected to the computer.

Processor

It displays the processor according to the selected board. It refreshes every time during the selection of the board.

Port

It consists of the virtual and real serial devices present on our machine.

Get Board Info

It gives the information about the selected board. We need to select the appropriate port before getting information about the board.

Programmer

We need to select the hardware programmer while programming the board. It is required when we are not using the onboard USB serial connection. It is also required during the burning of the Bootloader.

Burn Bootloader

The Bootloader is present on the board onto the microcontroller. The option is useful when we have purchased the microcontroller without the bootloader. Before burning the bootloader, we need to make sure about the correct selected board and port.

Help

When we click on the Help button on the Menu bar, a drop-down list will appear.

The Help section includes several documents that are easy to access, which comes along with the Arduino IDE. It consists of the number of options such as Getting Started, Environment, Troubleshooting, Reference, etc. We can also consider the image shown above, which includes all the options under the Help section.

Some documents like Getting started, Reference, etc., can be accessed without the internet connection as well. It will directly link us to the official website of Arduino.

SYNTAX

Arduino Coding Basics

Example: To light the LED connected to pin number 13. We want to ON the LED for 4 seconds and OFF the LED for 1.5 seconds.

Code:

```
void setup ()
{
```

```
pinMode ( 13, OUTPUT);  // to set the OUTPUT mode of pin number 13.
}
void loop ()
{
digitalWrite (13, HIGH);
    delay (4000);  // 4 seconds = 4 x 1000 milliseconds
digitalWrite (13, LOW);
    delay (1500);  // 1.5 seconds = 1.5 x 1000 milliseconds
}
```

LICENSING

It is a free software which is available on the www.arduino.cc website and it can be downloaded for different platforms like iOS, Windows, Linux etc..,

APPLICATIONS – ARDUINO IR KIT

This is a software which is available on the playstore for Android and available on AppStore in iOS which can be downloaded and installed in various mobile phones.

ARDUINO IR KIT FEATURES

-Replace all your remote controls with your phone.

- Create your own universal IR remote control with Arduino+Bluetooth shield on Module.
- Send and receive infrared signals with any protocols.
- Store your IR database on your phone.
- Auto-connect to Arduino

ARDUINO IR KIT DEPLOYMENT

This can be installed directly from Playstore for android or it can be downloaded in the form of .apk and can be installed through file manager

In iPhone systems install it from official AppStore available in default and it will be installed automatically.

GRAPHICAL USER INTERFACES

It has one of the best UI which supports various tabs listed below:

Home:

This contains all the newly inserted buttons and it contains the Group buttons assigned by the user manually.

It contains Menu bar on left top corner which contains Clear all options to clear the existing buttons and an option to Add group button.

It contains Sort bar on the right top corner which contains the options to rearrange the available buttons and its ID.

Settings:

This contains the option to select BLE Module, Board type, Auto connect option and some extra buttons like Connect, Search, Disconnect.

It also contains the option to IR receiver whether to enable it or not.

Info:

It holds some details like How to use, Contact us, Rate us and some details about its copyrights.

Connected/Disconnected:

This shows the current status whether the Bluetooth module is connected or not.

DATABASE ACCESS

This app specially contains the database storage and retrieval command on it which uses the mobile’s in-built storage to store the data of the buttons which are configured early and they can be used in further.

SYSTEM ANALYSIS

EXISTING SYSTEM

- IR based Remotes gets easily blocked by simple objects like notebooks,wall etc…..,
- Specific mobiles which contain IR transmitter can only be used for remote operation.
- Devices available similar to this contain only limited number of popular remotes supported.
- The storage and retrieval options are not available in previous versions.

PROPOSED SYSTEM

- The purpose of this tool is to control an IR remote handled device using a mobile phone using Bluetooth.
- This makes the controlling of any component easier and handy.
- The IR signals from remote to be used with actual device will be captured and stored .
- The previous remotes used can be stored using the provided mobile application.
- The remotes used can be named and configured easily using the application.

SYSTEM DESCRIPTION

INPUT DESIGN

Input design is the process of converting the user-oriented. Input to a computer based format. The goal of the input design is to make the data entry easier, logical and free error. Errors in the input data are controlled by the input design. The quality of the input determines the quality of the system output.

OUTPUT DESIGN

Output design is very important concept in the computerized system, without reliable output the user may feel the entire system is unnecessary and avoids using it. The proper output design is important in any system and facilitates effective decision-making. The output design of this system includes various reports.

CODE DESIGN

Code is an order collection of symbols designed to provide unique identification of an entry or attribute. Sometimes used in the place of name of the item they can be specified all object’s physical or performances characteristics or operational instructions.

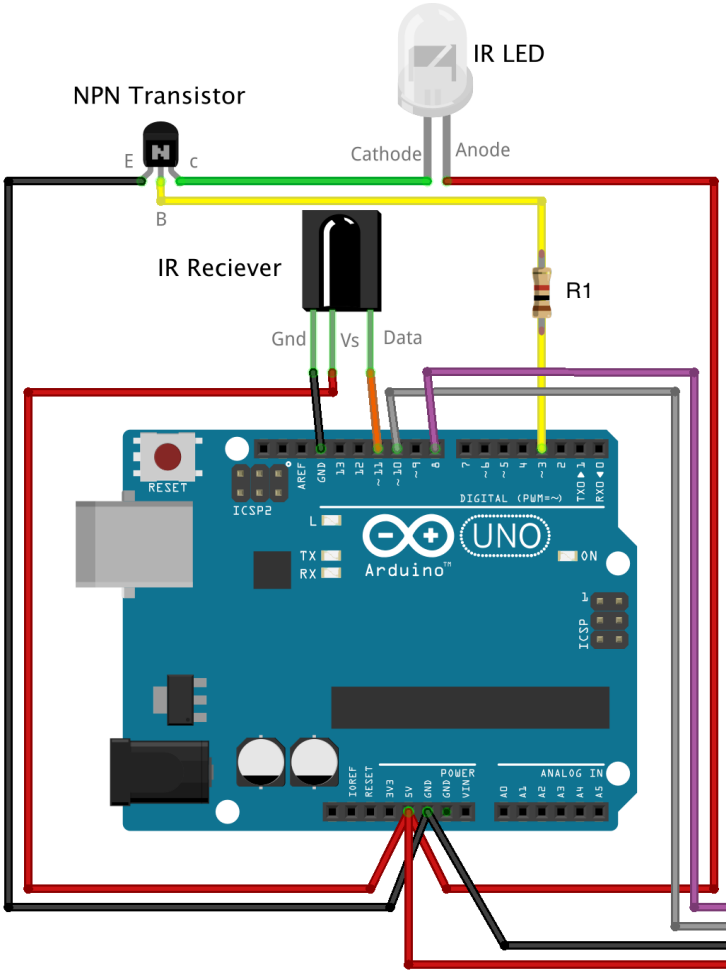
They can also show inter relationship and may sometime be used to achieve secrecy or confidentiality.

Most computer systems are stable from the compiler down to the execution of binary instructions. Therefore, it's natural to think of "product" as the artifact just above that base. That's language source code.

The system that produces that artifact is still quite unpredictable, so it's not likely we'll shift our orientation. Artifacts more abstract, be they whatever, will constitute a nebulous range of "designs" or "specifications".

DIAGRAMS

CIRCUIT DIAGRAM :



Components:
- Bluetooth Module (HC-06 or HC-05).
- IR Reciever (VS1838b or TSOP382 or equivalent).
- IR Transmitter (Any IR LED)
- NPN Transistor (C945 or any equivalent).

Notes:
1- HC-06 & HC-05 are working with Android only.
2- If you are using Arduino MEGA board then Connect the Yellow cable to pin 9.
3- If you are using Arduino Leonardo board then Connect the Yellow cable to pin 13.

TESTING

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

Functional test :

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

CODING

CODING :

```
/*
  Title  : Arduino IR
  version: V2.0
*/

/*
  Connection:
  arduino_rx_pin  -------->   Bluetooth_tx_pin
  arduino_tx_pin  -------->   Bluetooth_rx_pin
  - If you are using Bluefruit module then make sure to connect CTS pin to ground.
  Notes:
      - Make sure to install IRremote library from Sketch/Include Library/Manage Libraries then search for "IRremote" then install it.
*/
#include <SoftwareSerial.h>
#include <IRremote.hpp>

#define arduino_rx_pin 10  //  arduino_rx_pin 10 -------->   Bluetooth_tx_pin
#define arduino_tx_pin 8  //  arduino_tx_pin 8 -------->   Bluetooth_rx_pin

unsigned int irBuf[100];
unsigned int irBufLen;
unsigned int irBufType;
boolean repeat = false;

SoftwareSerial mySerial(arduino_rx_pin, arduino_tx_pin); // RX, TX
```

```
#define IR_RECEIVE_PIN  11 // Reciever Pin
#define IR_SEND_PIN     3  // Send Pin

IRData IRSendData;
bool rawFlag; //Recieving raw or encoded singnal

void setup(void)
{
  Serial.begin(9600);
  IrReceiver.begin(IR_RECEIVE_PIN, DISABLE_LED_FEEDBACK);
  IrSender.begin(IR_SEND_PIN, DISABLE_LED_FEEDBACK);
  Serial.println("Start");

   mySerial.begin(9600);//If you need to change this, then you have to change bluetooth module baudrate to the same.
}

void loop(void)
{
  if ( mySerial.available() )
  {
    process();
  }

  if (IrReceiver.decode()) {  // Grab an IR code
    dumpCode();           // Output the results as source code
    delay(50);
    IrReceiver.resume();              // Prepare for the next value
  }
```

```
  if (repeat) {
    rawFlag == true ? sendCode(1) : sendCode(0) ;//sent repeat raw signal or
encoded signal

  }

}

void process() {

  String command = mySerial.readStringUntil('/');

  if (command == "ir") {
   irCommand();
  }

  if (command == "irR") {//R for repeated code
   irCommandR();
  }

  if (command == "allstatus") {
   allstatus();
  }
}

void irCommand() {

  repeat = false;
```

```
String codeType, codeValue, codeLen;
codeType = mySerial.readStringUntil('/');
codeValue = mySerial.readStringUntil('/');
codeLen = mySerial.readStringUntil('\r');

if (codeType.toInt()) {//If encoded signal
  int commaIndex = codeValue.indexOf(',');
  int secondCommaIndex = codeValue.indexOf(',', commaIndex + 1);

    String codeAdress = codeValue.substring(0, commaIndex);//Commands like: mode, digital, analog, servo,etc...
        String codeCommand = codeValue.substring(commaIndex + 1, secondCommaIndex);//Pin number.
    String decodedRawData = codeValue.substring(secondCommaIndex + 1 );// value of the pin.
  // prepare data
  IRSendData.protocol = codeType.toInt();
  IRSendData.address = codeAdress.toInt();
  IRSendData.command = codeCommand.toInt();
  IRSendData.decodedRawData = decodedRawData.toInt();
  sendCode(0);
  //   Serial.println(F("Sent Encoded "));
} else {//If raw signal
  irBufLen = codeLen.toInt();
  irBufType = codeType.toInt();
  stringToIntArry(codeValue);
  sendCode(1);
  //   Serial.println(F("Sent Raw "));
}
```

```
}

void irCommandR() {

  String codeType, codeValue, codeLen;
  codeType = mySerial.readStringUntil('/');

  if (codeType == "off") {
    repeat = false;
    Serial.println(F("Don't repeat"));
  } else {
    codeValue = mySerial.readStringUntil('/');
    codeLen = mySerial.readStringUntil('\r');
    if (codeType.toInt()) {//If encoded signal
      int commaIndex = codeValue.indexOf(',');
      int secondCommaIndex = codeValue.indexOf(',', commaIndex + 1);

        String codeAdress = codeValue.substring(0, commaIndex);//Commands like: mode, digital, analog, servo,etc...
            String codeCommand = codeValue.substring(commaIndex + 1, secondCommaIndex);//Pin number.
        String decodedRawData = codeValue.substring(secondCommaIndex + 1 );// value of the pin.
      // prepare data
      IRSendData.protocol = codeType.toInt();
      IRSendData.address = codeAdress.toInt();
      IRSendData.command = codeCommand.toInt();
                                          IRSendData.decodedRawData = decodedRawData.toInt();//IRDATA_FLAGS_EMPTY
```

```
      repeat = true;
      rawFlag = false;
      //      Serial.println(F("Sent Repeat Encoded "));

    } else {//If raw signal

      stringToIntArry(codeValue);
      irBufLen = codeLen.toInt();
      irBufType = codeType.toInt();
      repeat = true;
      rawFlag = true;
      //      Serial.println(F("Sent Repeat Raw "));
    }
  }

}

void allstatus() {
  String data_status = "{\"T\":\"\",\"D\":[]}";
  mySerial.println(data_status);
  Serial.println(F("Connected"));
}

void  dumpCode ()
{
  int codeType = IrReceiver.decodedIRData.protocol;
  int codeAdress = IrReceiver.decodedIRData.address;
  int codeCommand = IrReceiver.decodedIRData.command;
  int codeExtra = IrReceiver.decodedIRData.extra;
```

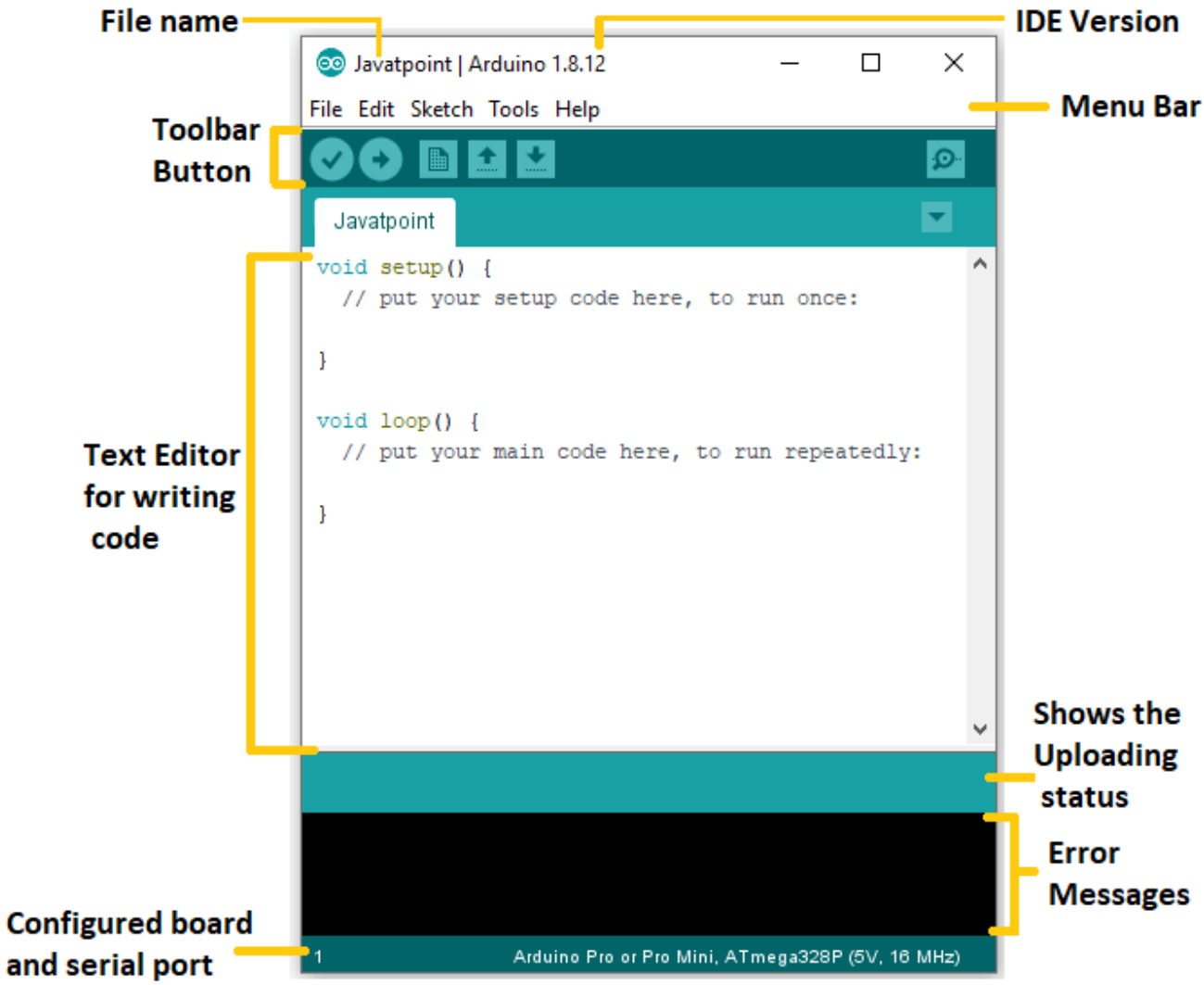
```
  int codeLen = IrReceiver.decodedIRData.rawDataPtr->rawlen;
  unsigned long codeValue = IrReceiver.decodedIRData.decodedRawData;
    //    const uint16_t codeBuf = IrReceiver.decodedIRData.rawDataPtr-
>rawbuf;
  String data_status;
  data_status += F("{\"T\":\"");
  data_status += codeType;
  data_status += F("\",\"D\":[\"");
  data_status += codeValue;
  data_status += F("\",\"");
  if (codeType) {
    data_status += codeAdress;
    data_status += F(",");
    data_status += codeCommand;
    data_status += F(",");
    data_status += codeValue;
  }
  else {
#if RAW_BUFFER_LENGTH <= 254         // saves around 75 bytes program
space and speeds up ISR
    uint8_t i;
#else
    uint16_t i;
#endif
    for (int i = 1; i < codeLen ; i++)
    {
      if (i & 1) {
        // Mark
```

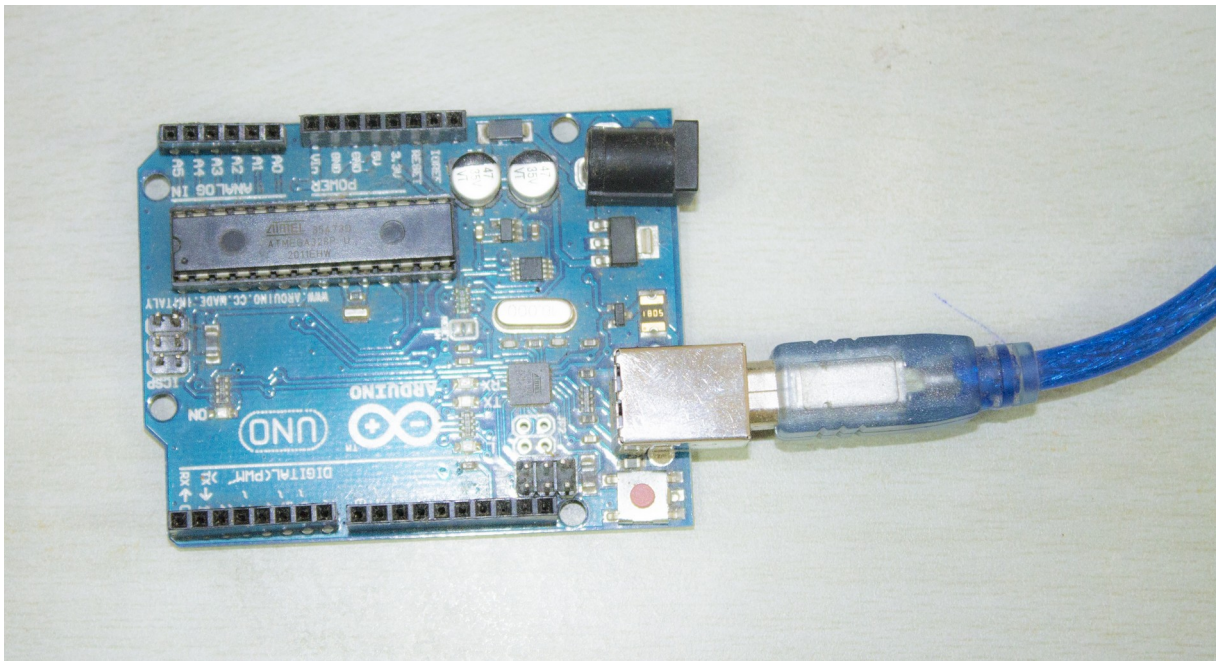
```
            data_status  +=  IrReceiver.decodedIRData.rawDataPtr->rawbuf[i]  *
MICROS_PER_TICK - MARK_EXCESS_MICROS;
     } else {
            data_status  +=  IrReceiver.decodedIRData.rawDataPtr->rawbuf[i]  *
MICROS_PER_TICK + MARK_EXCESS_MICROS;
     }
     if (i != codeLen - 1 )data_status += ",";
   }
  }

  data_status += F("\",\"");
  data_status += codeLen;
  data_status += F("\"]}");
  mySerial.println(data_status);
  //  Serial.println(F("Got IR Code"));
  //  Serial.println(data_status);
  IrReceiver.printIRResultShort(&Serial);
}


void sendCode(int x) {
  if (x == 0) {//If encoded signal
    IrSender.write(&IRSendData, 2);
    delay(50);
    IrReceiver.resume();

  }

  if (x == 1) {//If raw signal
```

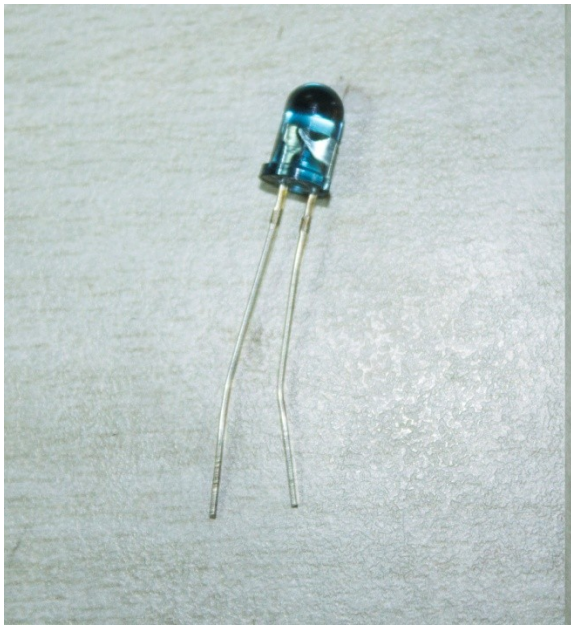
```
    IrSender.sendRaw(irBuf, irBufLen, 38);
    delay(50);
    IrReceiver.resume();

  }
}

void stringToIntArry(String irRawString) {
  String dataShort = "";
  int counter = 0;
  for (int i = 0; i <= irRawString.length(); i++) {
    dataShort += irRawString[i];
    if (irRawString[i] == ',') {
      dataShort = dataShort.substring(0, dataShort.length() - 1 );
      irBuf[counter] = dataShort.toInt();
      counter = counter + 1;
      dataShort = "";
    }
  }
}
```

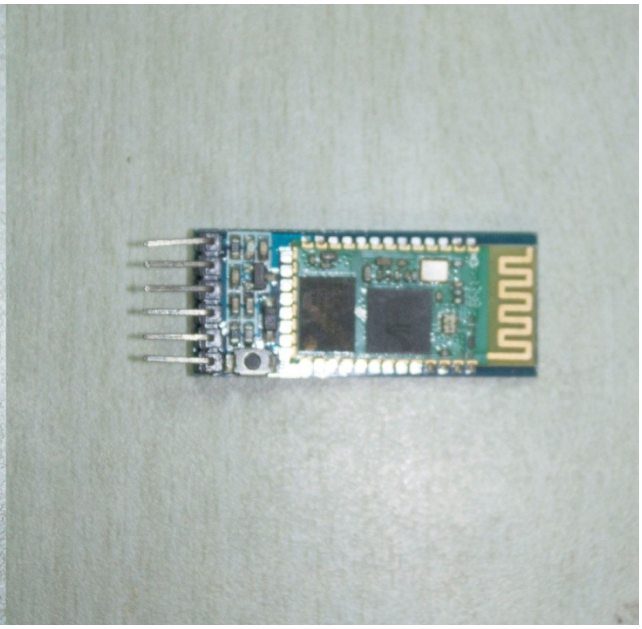
SCREENSHOTS

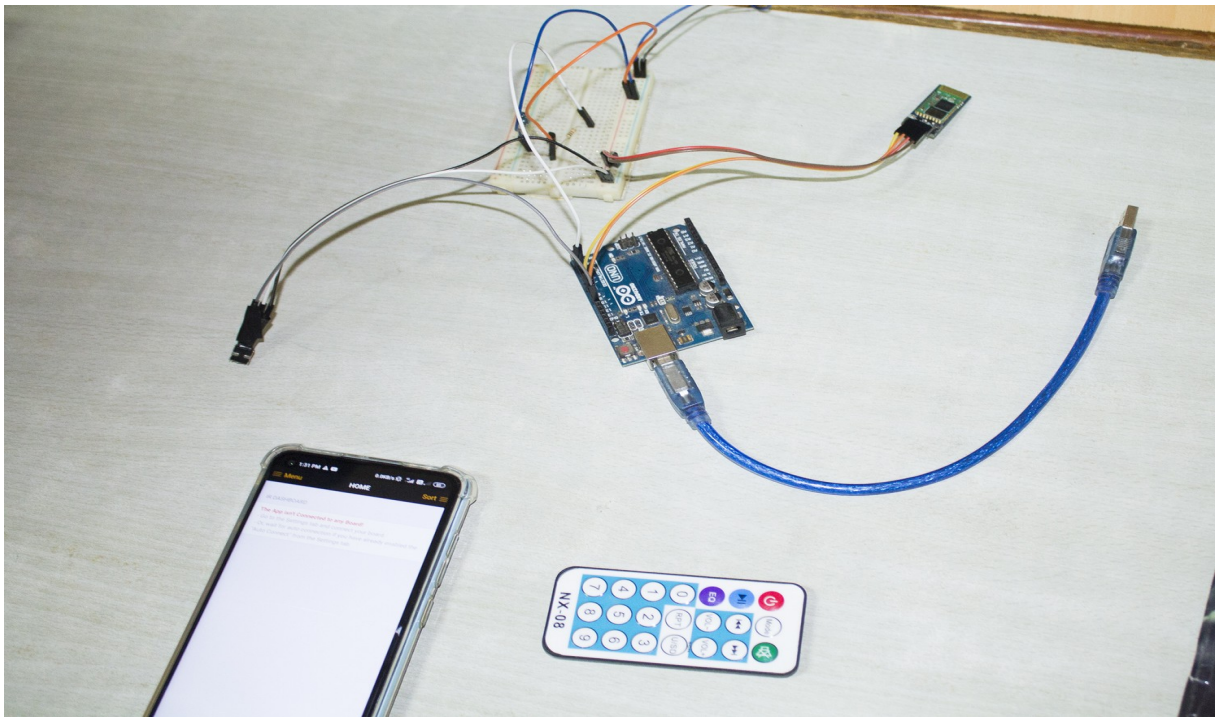
SCREENSHOTS

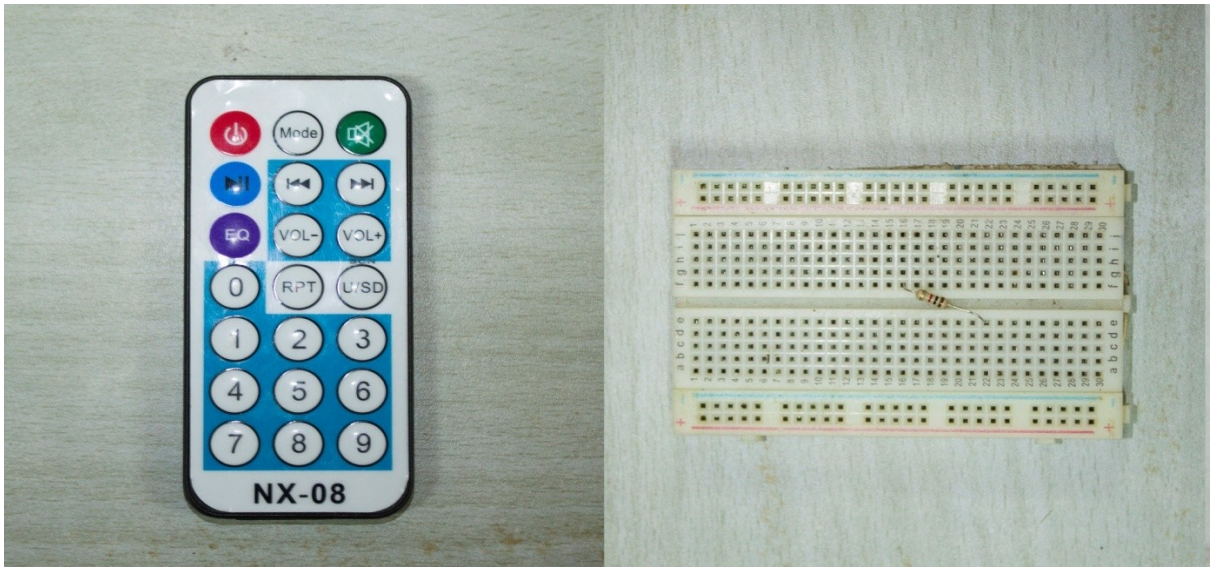


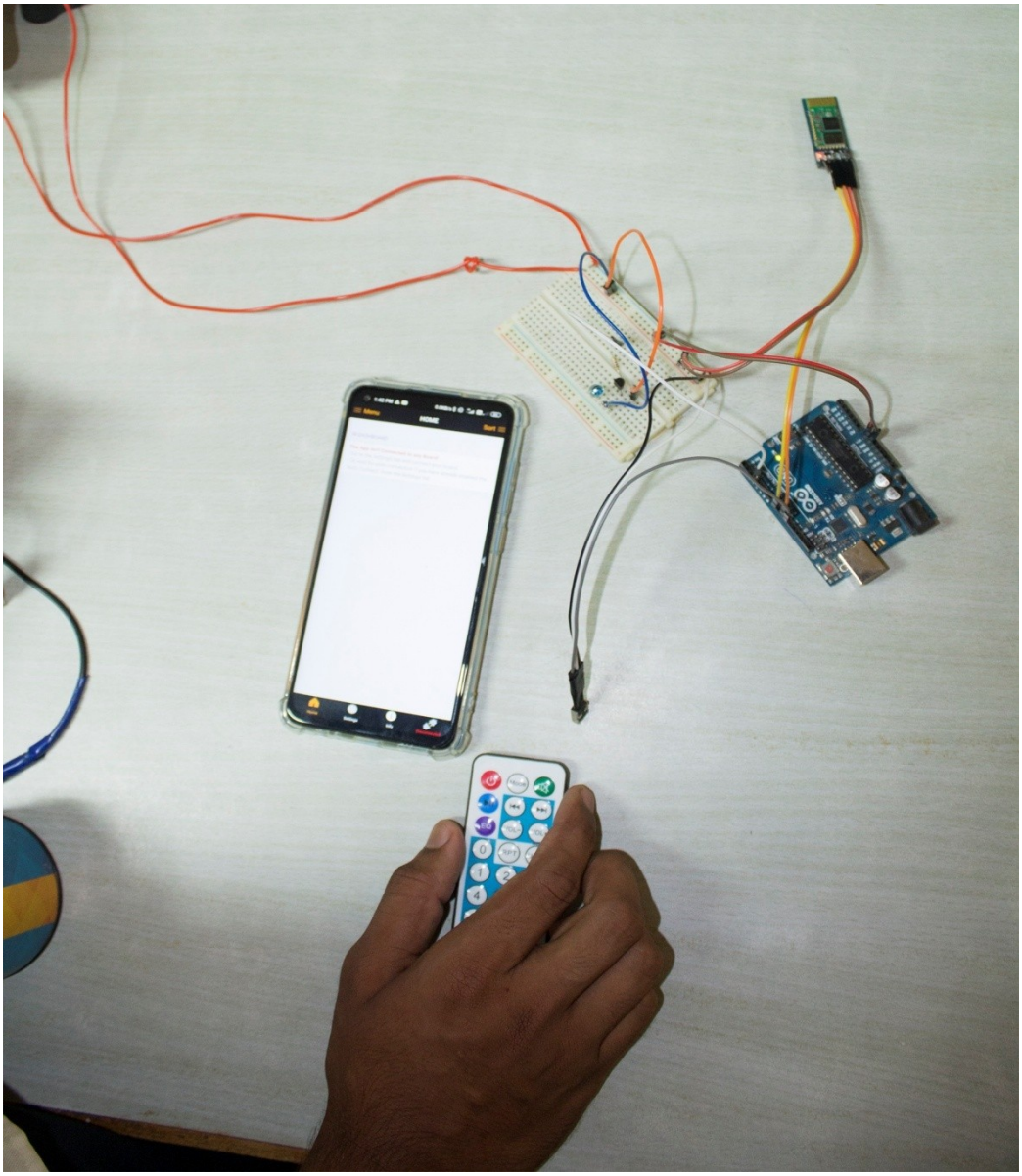

UNO
ARDUINO








Mode
VOL-
VOL+
EQ
RPT
0
1
2
3
4
5
6
7
8
9
NX-08



\

CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION :

- Reduces the difficulties of using an ordinary IR remote.
- Multiple remotes and maintenance is avoided.

FUTURE ENHANCEMENT:

- Auto configuring of remotes using data from internet.
- Data’s can be published on internet for further usage.
- Voice based controlling of components.

REFERENCES

REFRENCES

1. **Arduino IDE – Arduino CC**
2. **Arduino IR Kit – Tacto IR**
3. **Basic Electronic Components – by Edwin Dayand**