**6.1 Initial Setups in Arduino IDE Software**

**Step 1:** Install the Arduino 1.6.7 IDE.

**Step 2:** Go to File>>Preferences>>Additional Boards Manager URLs:

http://arduino.esp8266.com/stable/package\_esp8266com\_index.json

**>>**Ok

**Step 3:** Go to Tools>>Boards>>Boards Manager>>

Download the **“esp8266** by **ESP8266 Community** version **2.2.0”**

**Step 4:** Go to Tools>>Boards>>Generic ESP8266 Module

**Step 5:** Go to Tools>>Upload Speed>>115200

Port>>choose preferred COM ports.

**6.2 How to Flash ESP8266-12**

**Step 1:** First upload BareMinimum code to Arduino Uno board.

**Step 2:** Connect the pins of Arduino Uno to ESP8266-12 pins as mention below Arduino Pins  ESP8266 pins

* 5v  5v
* 3.3v  3.3v, CH\_PD
* Gnd  Gnd (both)
* Tx  Tx
* Rx  Rx

**Note:** Program code is directly uploaded into ESP8266 module. In this case, Arduino board is used as a Flash Burner, i.e. code is directly uploaded to ESP8266 module.

**Step 3:** Reset the ESP8266 by connecting RESET pin to GND3.3vGND and disconnect. But Explore ESP8266 Wi-Fi module has inbuilt Reset button. Press the reset button to reset the module.

**Step 4:** While uploading the program code connect the GPIO 0 to GND.

**Step 5:** Once upload is successful, disconnect the GPIO 0 from GND.

**6.3 Experimental Setup**

**6.3.1 List of Components**

**Components Quantity**

Explore ESP8266 Wi-Fi Module 1

Relay Switch 1

Power Supply –

5v 1 (from Arduino)

3.3v 1 (from Arduino)

Gnd From Arduino

**6.3.2 Circuit Connection Procedure**



**6.3.3 Physical Connection**

Physical connections include the placing the sensors and actuators in small model of agriculture field and includes proper connections. Detail is given below

* The field includes two Regions: 1, 2 and a water reservoir.
* Submerse the submersible motor pump in the reservoir.
* Place the valve 1 in Region 1.
* Place the valve 2 in Region 2.
* Make proper pipeline connections from motor pump to the valves.
* Extend the pipeline connection to respective fields to supply water.
* Make some arrangements to supply the water like making the holes to pipes.
* Place the water level indicator in the water reservoir.
* Place the soil moisture sensor 1 in Region 1 near the roots of the plants.
* Place the soil moisture sensor 2 in Region 2 near the roots of the plants.

Give all required supply voltages.

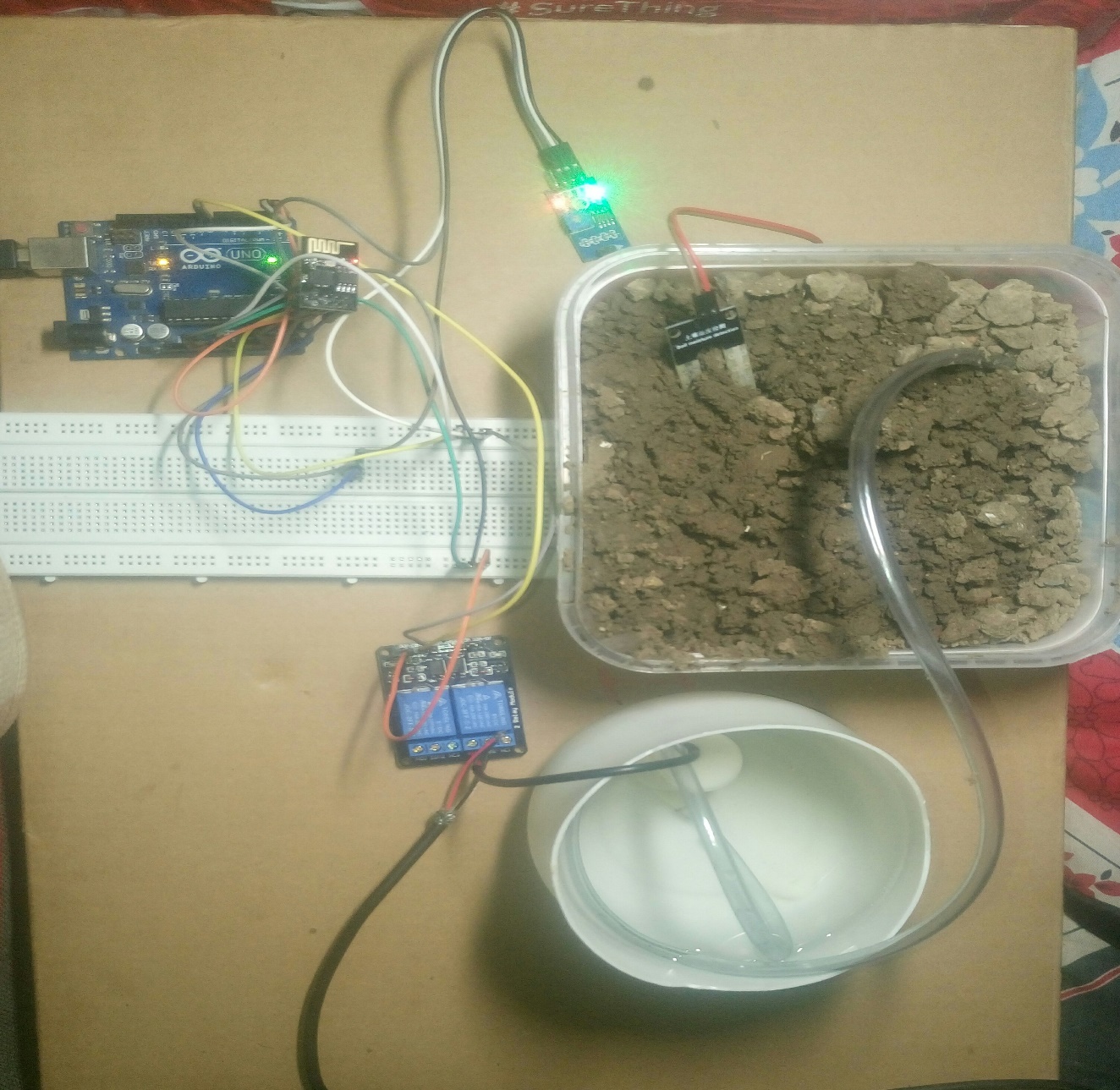


Figure 6.3 project setup

**6.4 Results**

**6.4.1 How Module Works?**

The installation of the irrigation monitoring and controlling system using IoT is done. It is found that the system works properly and the water is passed to the plants as when required. If the soil is dry, an alert message is sent” MOTOR ON” to the mobile and through text, water pump starts which leads to water to flow. If the soil is wet, an alert message is sent” MOTOR ON” to the mobile and through text, the water pump is turned off and water flow stop. We have used an android application i.e. Blue term. These applications work totally on wi-fi module. To interface the android application and the master robot we require wi-fi module. The application Blue term is used for coding and writing programming instructions and this programming data is sent via wi-fi through the internet. This application acts like an emulator which then is given as the input to the microcontroller Arduino Uno. This Set of codes is then given to the input of the motor driver which is responsible for the movement of the motor. As a result of which the Motor will start and water will be supplied to plants. The same codes are simultaneously sent to the output pin of the microcontroller.

*Table 1*

*MOISTURE REQUIREMENTS*

|  |  |
| --- | --- |
| *Moisture*  *level* | *Motor*  *Status* |
| *1020* | *ON* |
| *850* | *ON* |
| *220* | *OFF* |
| *300* | *ON* |
| *32* | *OFF* |
| *245* | *OFF* |

**6.4.2 Controlling the Module using MyMQTT Android App**

MyMQTT is a simple Message Queue Telemetry Transport (MQTT) client for

Android. The service is being provided by **Mosquitto** of **eclipse** a service provider.

**Features:** Connect to MQTT v3.1 Broker (optional with username and password) Subscribe to various topics.

Publish messages to a topic

Save messages

Filter received Messages

**Code:-**

/\*

-- New project --

This source code of graphical user interface has been generated automatically by

RemoteXY editor.

To compile this code using RemoteXY library

2.3.3 or later version

download by link http://remotexy.

com/en/library/

To connect using RemoteXY mobile app by

link http://remotexy.

com/en/download/

- for ANDROID 4.1.1 or later version;

- for iOS 1.2.1 or later version;

This source code is free software; you can

redistribute it and/or

modify it under the terms of the GNU Lesser

General Public

License as published by the Free Software

Foundation; either

version 2.1 of the License, or (at your

option) any later version.

\*/

//////////////////////////////////////////////

// RemoteXY include library //

//////////////////////////////////////////////

// RemoteXY select connection mode and include

library

#defineREMOTEXY\_MODE\_\_ESP8266\_HARDSERIAL\_POINT

#include <RemoteXY.h>

// RemoteXY connection settings

#define REMOTEXY\_SERIAL Serial

#define REMOTEXY\_SERIAL\_SPEED 115200

#define REMOTEXY\_WIFI\_SSID "RemoteXY"

#define REMOTEXY\_WIFI\_PASSWORD "12345678"

#define REMOTEXY\_SERVER\_PORT 6377

// RemoteXY configurate

#pragma pack(push, 1)

uint8\_t RemoteXY\_CONF[] =

{ 255,2,0,4,0,58,0,8,13,0,

2,1,9,23,22,11,2,26,31,31,

79,78,0,79,70,70,0,1,0,73,

23,12,12,2,31,88,0,66,130,45,

8,13,10,16,26,65,4,38,24,9,

9,65,2,56,24,9,9,66,0,48,

37,7,16,2,26 };

struct {

// input variable

uint8\_t switch\_1; // =1 if switch ON and =0

if OFF

// output variable

uint8\_t led\_1\_r; // =0..255 LED Red

brightness

uint8\_t led\_1\_g; // =0..255 LED Green

brightness

int8\_t level\_2; // =0..100 level position

// other variable

uint8\_t connect\_flag; // =1 if wire

connected, else =0

} RemoteXY;

#pragma pack(pop)

/////////////////////////////////////////////

// END RemoteXY include //

/////////////////////////////////////////////

#define PIN\_SWITCH\_1 13

#define PIN\_SWITCH\_2 11

int sensorPin = A2; // select the input pin

for the potentiometer

// select the pin for the LED

int sensorValue = 0;

void setup()

{

RemoteXY\_Init ();

Serial.begin(9600);

pinMode (PIN\_SWITCH\_1, OUTPUT);

// TODO you setup code

}

void loop()

{

RemoteXY\_Handler ();

sensorValue = analogRead(sensorPin);

RemoteXY.level\_2=sensorValue;

if(sensorValue > 250)

{

digitalWrite(PIN\_SWITCH\_1,HIGH);

digitalWrite(PIN\_SWITCH\_2,HIGH);

RemoteXY.led\_1\_g=255;

}

else

{

digitalWrite(PIN\_SWITCH\_1,LOW);

digitalWrite(PIN\_SWITCH\_2,LOW);

RemoteXY.led\_1\_r=255;

}

//digitalWrite(PIN\_SWITCH\_1, (RemoteXY.

switch\_1==0)?LOW:HIGH);

//digitalWrite(PIN\_SWITCH\_1, (RemoteXY.

switch\_1==0)?LOW:HIGH);

// TODO you loop code

// use the RemoteXY structure for data

transfer

}