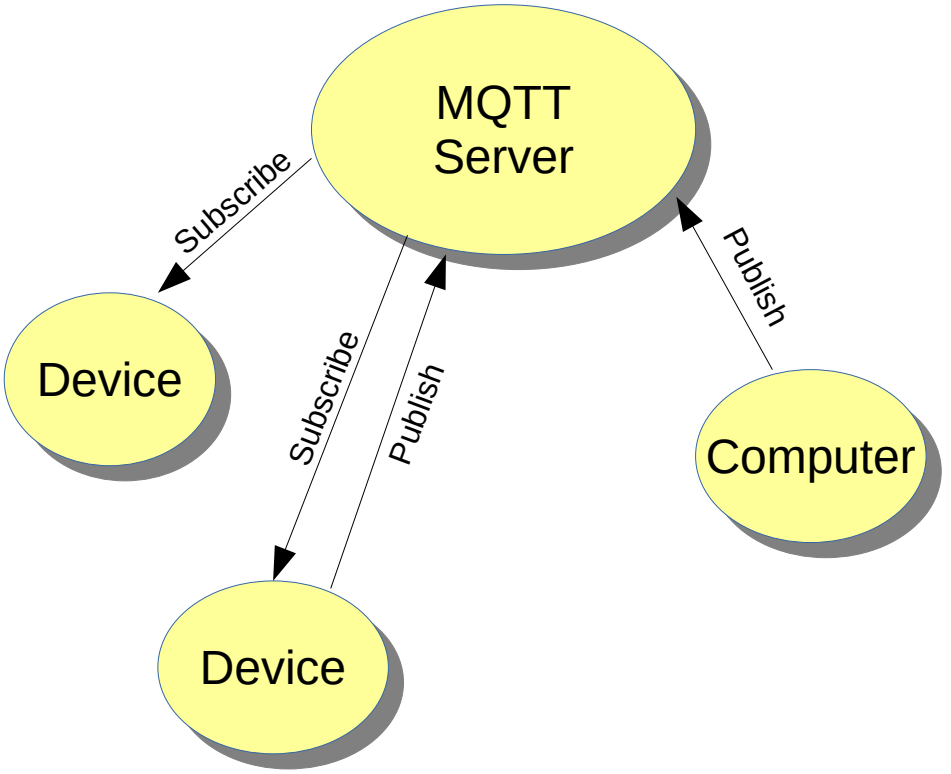
MQTT

Message Queuing Telemetry Transport

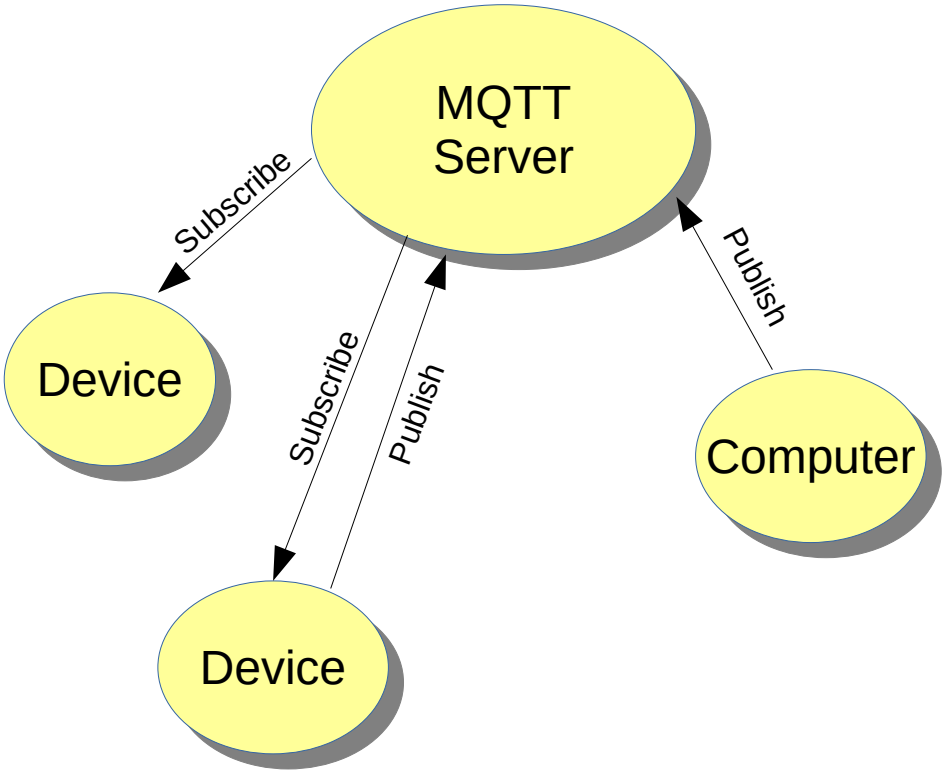
MQTT

- MQTT is a service that allows devices and systems to send and receive data.
- Data is passed using a publish and subscription model. Devices can send data by publishing to topics.
- Devices can subscribe to topics and receive any data published to that topic.



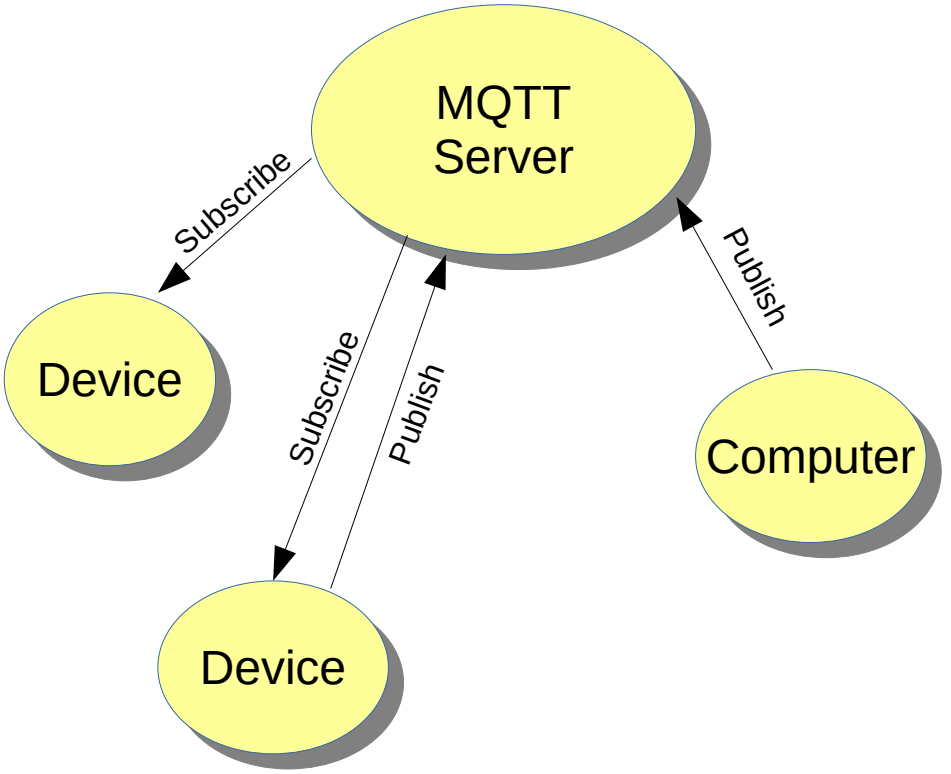
MQTT

- Passing data between devices are through topics. There can be many different topics. To receive data, a device subscribes to a topic.
- To control a device through MQTT another device or computer publishes data through the same topic.



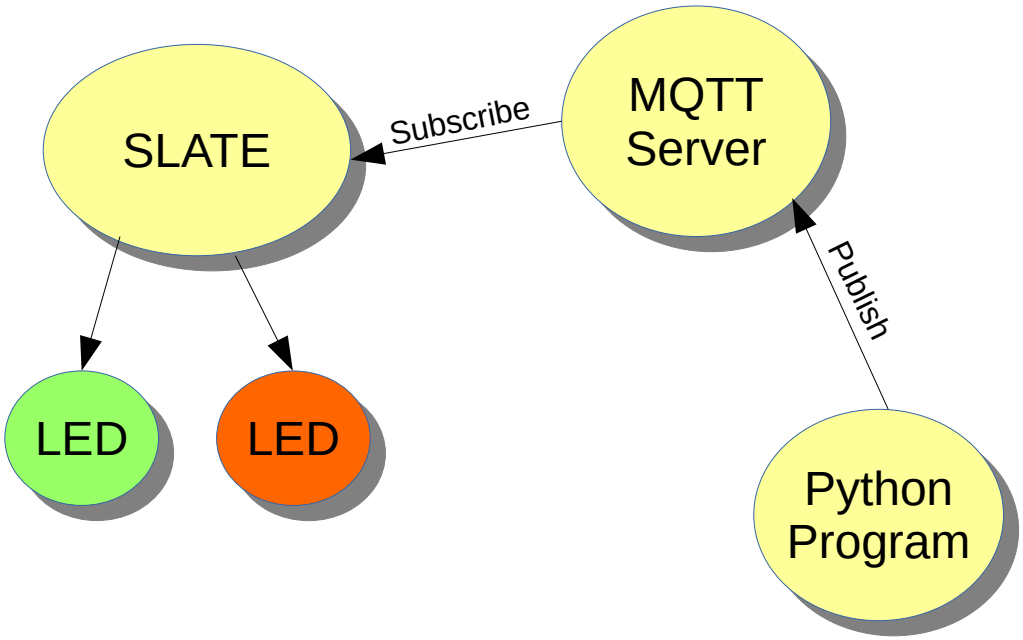
MQTT

- Any number of devices can subscribe or publish to the same topic. If several devices subscribe to the same topic, any data published to that topic will be sent to all the devices.



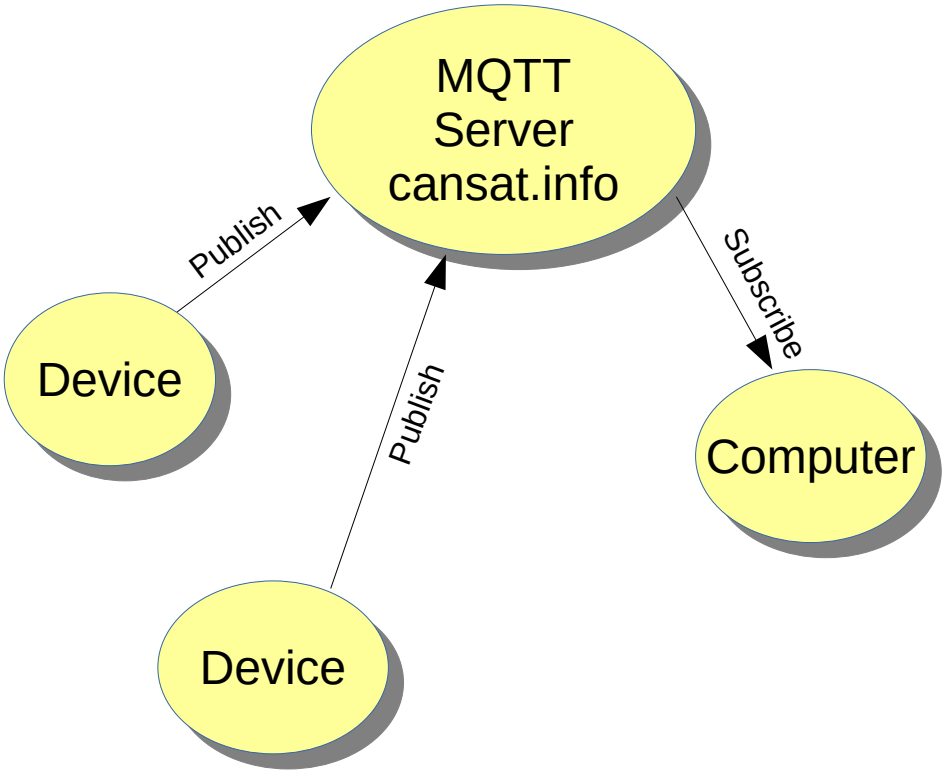
MQTT Example

- In this example, the SLATE will be set up to subscribe to the topic LED. The data sent to the topic will indicate which LED to turn on or off.
- A python program will be used to turn the LEDs on and off.
- The message or data to be sent from the python program will be two text characters. The first turns the red LED on and off by using 1 and 0. The second character controls the green LED.



MQTT 2

- The mqtt service resides on a computer at **cansat.info**. A user name and password is required.
- The user name is **gmu2021**
- The password is **rockets41!**



PubSubClient for the SLATE

- An mqtt library is needed for the SLATE. The PubSubClient library works well.
- On the Arduino IDE, select the **Sketch** menu and the **Include Library** submenu. Select **Manage Libraries.**
- A window pops open with a list of available libraries. Enter **pubsubclient** in the top right search filter text entry.
- Scroll, locate and install **PubSubClient**.

MQTT Client for Python

- Python will need the Paho-mqtt client module.
- Open a terminal or Power shell. Enter the following:

  pip3 install paho-mqtt
- The module should get installed.

SLATE MQTT Subscribe

- The SLATE will subscribe and turn LEDs on and off.
- Connect the red LED to digital pin 13.
- Connect the green LED to digital pin 15.

SLATE MQTT Subscribe

- The code will connect to the mqtt server and then subscribe to topic LED.
- First, the include files are placed at the top of the program.
- The WiFi client is declared as esp.
- The mqtt client is defined as client and attached to the WiFi client. This makes all the mqtt communications through the WiFi.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

WiFiClient esp;
PubSubClient client(esp);

void setup() {
  Serial.begin(115200);
  pinMode(13,OUTPUT);
  pinMode(15,OUTPUT);
  WiFi.mode(WIFI_STA);
  WiFi.begin("SSID","PASSWORD");
  while(WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  randomSeed(micros());
  String clientID = "me" + String(random(0xffff));
  Serial.println();
  Serial.println(WiFi.localIP());
  client.setServer("cansat.info",1883);
  client.setCallback(callback);
  if(client.connect(clientID.c_str(),"gmu2021",rockets41!)) {
      Serial.println("Connected");
  }
  client.subscribe("gmu/LED");
}
```

SLATE MQTT Subscribe

- In the **setup()** function, the digital pins 13 and 15 are configured as ouputs.
- The SLATE attempts to connect to the WiFi network.
- Replace SSID with your own SSID.
- Replace PASSWORD with the WiFi router password.
- The SLATE will check if it connects every 500 ms.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

WiFiClient esp;
PubSubClient client(esp);

void setup() {
  Serial.begin(115200);
  pinMode(13,OUTPUT);
  pinMode(15,OUTPUT);
  WiFi.mode(WIFI_STA);
  WiFi.begin("SSID","PASSWORD");
  while(WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  randomSeed(micros());
  String clientID = "me" + String(random(0xffff));
  Serial.println();
  Serial.println(WiFi.localIP());
  client.setServer("cansat.info",1883);
  client.setCallback(callback);
  if(client.connect(clientID.c_str(),"gmu2021",rockets41!)) {
      Serial.println("Connected");
  }
  client.subscribe("gmu/LED");
}
```

SLATE MQTT Subscribe

- Next, a random seed is generated for the random number generator.
- A clientID is created for connecting to the mqtt server. Each connection needs a unique client ID. Using a random number generator to create a unique client ID works well.
- If a client connects with the same ID, the previous client will get disconnected.
- In the function **setserver()**, replace laptopIP with the IP address recorded.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

WiFiClient esp;
PubSubClient client(esp);

void setup() {
  Serial.begin(115200);
  pinMode(13,OUTPUT);
  pinMode(15,OUTPUT);
  WiFi.mode(WIFI_STA);
  WiFi.begin("SSID","PASSWORD");
  while(WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  randomSeed(micros());
  String clientID = "me" + String(random(0xffff));
  Serial.println();
  Serial.println(WiFi.localIP());
  client.setServer("cansat.info",1883);
  client.setCallback(callback);
  if(client.connect(clientID.c_str(),"gmu2021",rockets41!)) {
      Serial.println("Connected");
  }
  client.subscribe("gmu/LED");
}
```

SLATE MQTT Subscribe

- A callback function is set up with the **setCallback()** function. Any time a message is received through the subscribed topic, the callback function will be called.
- Next, the connection is made using the client ID. The **.c_str()** attached to **clientID** is used to convert the string variable to a byte array for the function.
- After connecting, the **subscribe()** function subscribes to **gmu/LED**.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

WiFiClient esp;
PubSubClient client(esp);

void setup() {
  Serial.begin(115200);
  pinMode(13,OUTPUT);
  pinMode(15,OUTPUT);
  WiFi.mode(WIFI_STA);
  WiFi.begin("SSID","PASSWORD");
  while(WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  randomSeed(micros());
  String clientID = "me" + String(random(0xffff));
  Serial.println();
  Serial.println(WiFi.localIP());
  client.setServer("cansat.info",1883);
  client.setServer("cansat.info",1883);
  client.setCallback(callback);
  if(client.connect(clientID.c_str(),"gmu2021",rockets41!)) {
      Serial.println("Connected");
  }
  client.subscribe("gmu/LED");
}
```

SLATE MQTT Subscribe

- The callback function has three arguments.
- The first argument points to the topic for the message received.
- The second argument is the data being passed.
- The third argument specifies the length of the data or number of bytes.

```
void callback(char *topic,byte* payload, uint16_t length) {
  if(!strcmp(topic,"gmu/LED")) {
    if(payload[0] == '1') digitalWrite(13,HIGH);
    else if(payload[0] == '0') digitalWrite(13,LOW);
    if(payload[1] == '1') digitalWrite(15,HIGH);
    else if(payload[1] == '0') digitalWrite(15,LOW);
  }
}

void loop()
{
  client.loop();
}
```

SLATE MQTT Subscribe

- The first thing done is to determine if the data is from the topic gmu/LED. It is possible to subscribe to more than one topic. The same callback function is called for any subscribed topic.
- If more than one topic is subscribed, the topic has to be determined in the callback function.
- The **strcmp()** function will return zero if there is a match between topic and the text in quotes. This is why the exclamation point is used in front of **strcmp()**.

```
void callback(char *topic,byte* payload, uint16_t length) {
  if(!strcmp(topic,"gmu/LED")) {
    if(payload[0] == '1') digitalWrite(13,HIGH);
    else if(payload[0] == '0') digitalWrite(13,LOW);
    if(payload[1] == '1') digitalWrite(15,HIGH);
    else if(payload[1] == '0') digitalWrite(15,LOW);
  }
}

void loop()
{
  client.loop();
}
```

SLATE MQTT Subscribe

- The payload will consist of two text characters that are either 1 or 0.
- The first character turns the red LED on and off and the second character turns the green LED on and off.
- In the **loop()** function, the **client.loop()** is called repeatedly. This is what checks for any data received through the subscribed topics.

```
void callback(char *topic,byte* payload, uint16_t length) {
  if(!strcmp(topic,"gmu/LED")) {
    if(payload[0] == '1') digitalWrite(13,HIGH);
    else if(payload[0] == '0') digitalWrite(13,LOW);
    if(payload[1] == '1') digitalWrite(15,HIGH);
    else if(payload[1] == '0') digitalWrite(15,LOW);
  }
}

void loop()
{
  client.loop();
}
```

Python Publishing Program

- The python program will create a GUI with buttons to turn the LEDs on and off.

Python Publishing Program

- At the top of the program, import the tkinter module and the mqtt client module.
- Next, create an instance of the client called **mqttc**.
- **mqttc.connect()**, connects to the mqtt server. cansat.info runs the server.
- 1883 is the default port number for the mqtt server.

```
from tkinter import *
import paho.mqtt.client as mqtt

mqttc = mqtt.Client()
mqttc = username_pw_set('gmu2021','rockets41!')
mqttc.connect("cansat.info",1883)

redled = "0"
greenled = "0"

def redon():
    global redled,greenled
    print("Red ON")
    redled = "1"
    msg = redled + greenled
    mqttc.publish("gmu/LED",msg)

def redoff():
    global redled,greenled
    print("Red OFF")
    redled = "0"
    msg = redled + greenled
    mqttc.publish("gmu/LED",msg)
```

Python Publishing Program

- Next, two global variables are created to hold the state of the LEDs. One is for the red LED and the other for the green LED.
- The first function is for turning the red LED on. This function is called by the button that will be created to turn on the LED.

```
from tkinter import *
import paho.mqtt.client as mqtt

mqttc = mqtt.Client()
mqttc = username_pw_set('gmu2021','rockets41!')
mqttc.connect("cansat.info",1883)

redled = "0"
greenled = "0"

def redon():
    global redled,greenled
    print("Red ON")
    redled = "1"
    msg = redled + greenled
    mqttc.publish("gmu/LED",msg)

def redoff():
    global redled,greenled
    print("Red OFF")
    redled = "0"
    msg = redled + greenled
    mqttc.publish("gmu/LED",msg)
```

Python Publishing Program

- In the function, the **redled** and **greenled** variables are declared as globals. If this is not done, the function will declare them as local variables and not use the variables declared above.
- The **redled** variable is set to text value **1**.
- The variable **msg** is created and combines **redled** and **greenled** together.
- **mqttc.publish()** sends the contents of msg to the topic LED on the mqtt server.

```
from tkinter import *
import paho.mqtt.client as mqtt

mqttc = mqtt.Client()
mqttc = username_pw_set('gmu2021','rockets41!')
mqttc.connect("cansat.info",1883)

redled = "0"
greenled = "0"

def redon():
    global redled,greenled
    print("Red ON")
    redled = "1"
    msg = redled + greenled
    mqttc.publish("gmu/LED",msg)

def redoff():
    global redled,greenled
    print("Red OFF")
    redled = "0"
    msg = redled + greenled
    mqttc.publish("gmu/LED",msg)
```

Python Publishing Program

- The **redoff()** function is identical to the **redon()** with the exception that redled is set to text value **0**.

```
from tkinter import *
import paho.mqtt.client as mqtt

mqttc = mqtt.Client()
mqttc = username_pw_set('gmu2021','rockets41!')
mqttc.connect("cansat.info",1883)

redled = "0"
greenled = "0"

def redon():
    global redled,greenled
    print("Red ON")
    redled = "1"
    msg = redled + greenled
    mqttc.publish("gmu/LED",msg)

def redoff():
    global redled,greenled
    print("Red OFF")
    redled = "0"
    msg = redled + greenled
    mqttc.publish("gmu/LED",msg)
```

Python Publishing Program

- The functions for controlling the green LED are the same as the previous two functions with the exception of modifying the **greenled** variable.

```
def greenon():
    global redled,greenled
    print("Green ON")
    greenled = "1"
    msg = redled + greenled
    mqttc.publish("gmu/LED",msg)

def greenoff():
    global redled,greenled
    print("Green OFF")
    greenled = "0"
    msg = redled + greenled
    mqttc.publish("gmu/LED",msg)

rb1 = Button(text="RED ON",command=redon, width=20)
rb2 = Button(text="RED OFF",command=redoff,width=20)
gb1 = Button(text="GREEN ON",command=greenon,width=20)
gb2 = Button(text="GREEN OFF",command=greenoff,width=20)
rb1.pack()
rb2.pack()
gb1.pack()
gb2.pack()
mainloop()
```

Python Publishing Program

- This section of the code creates four buttons. Each button is associated with a specific function.
- The **width=20** parameter sets the width of the button fixed to 20 characters wide.

```
def greenon():
    global redled,greenled
    print("Green ON")
    greenled = "1"
    msg = redled + greenled
    mqttc.publish("gmu/LED",msg)

def greenoff():
    global redled,greenled
    print("Green OFF")
    greenled = "0"
    msg = redled + greenled
    mqttc.publish("gmu/LED",msg)

rb1 = Button(text="RED ON",command=redon, width=20)
rb2 = Button(text="RED OFF",command=redoff,width=20)
gb1 = Button(text="GREEN ON",command=greenon,width=20)
gb2 = Button(text="GREEN OFF",command=greenoff,width=20)
rb1.pack()
rb2.pack()
gb1.pack()
gb2.pack()
mainloop()
```

Python Publishing Program

- At the bottom of the program, the buttons are packed into a vertical stack.
- **mainloop()** starts the loop where the buttons are monitored and the associated functions are called.

```
def greenon():
    global redled,greenled
    print("Green ON")
    greenled = "1"
    msg = redled + greenled
    mqttc.publish("gmu/LED",msg)

def greenoff():
    global redled,greenled
    print("Green OFF")
    greenled = "0"
    msg = redled + greenled
    mqttc.publish("gmu/LED",msg)

rb1 = Button(text="RED ON",command=redon, width=20)
rb2 = Button(text="RED OFF",command=redoff,width=20)
gb1 = Button(text="GREEN ON",command=greenon,width=20)
gb2 = Button(text="GREEN OFF",command=greenoff,width=20)
rb1.pack()
rb2.pack()
gb1.pack()
gb2.pack()
mainloop()
```

Starting it Up

- With the MQTT server running, load the SLATE program and let it start.
- Start the python program.
- A window should open with four buttons.
- Try turning the LEDs on and off.