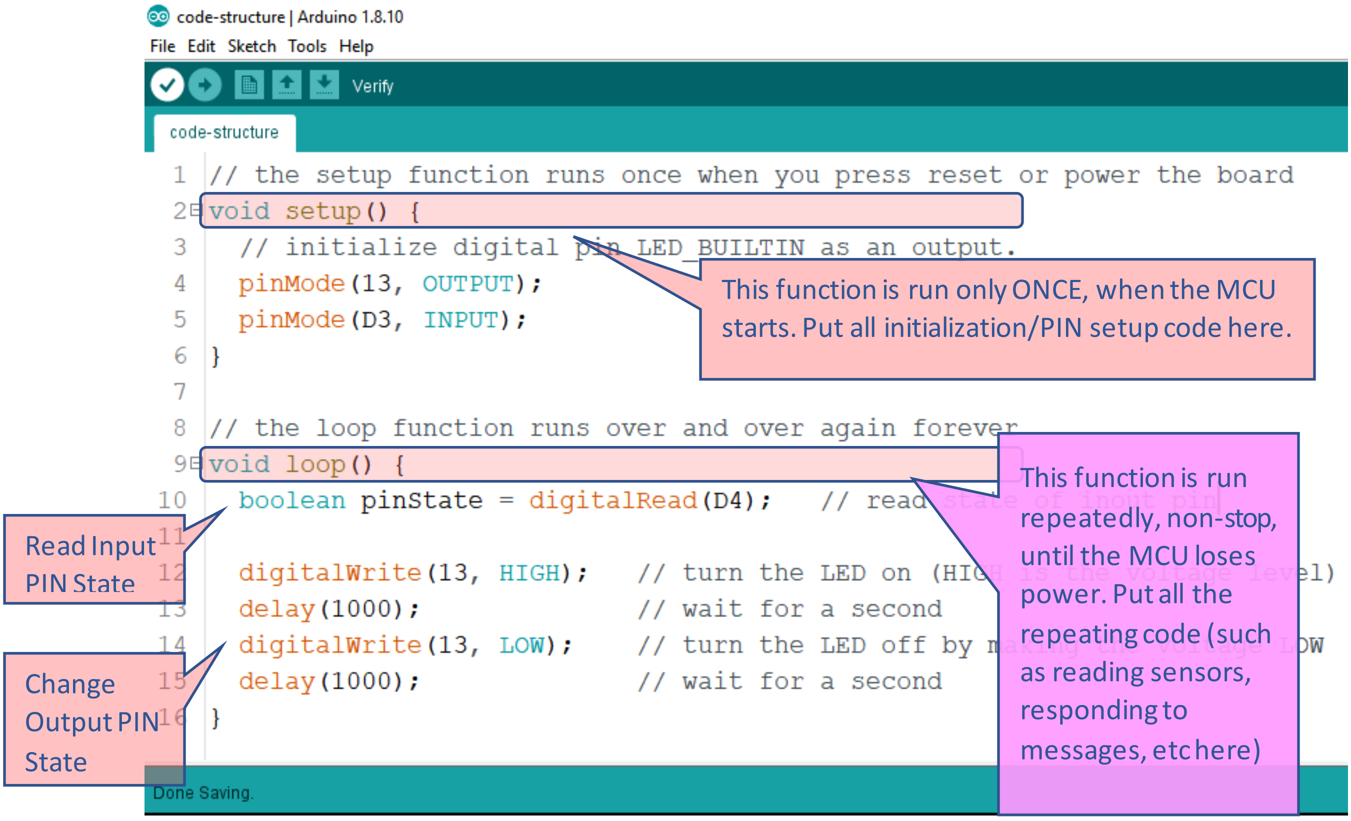
Structure of an embedded C Program

(For NodeMCU IoT Development)

This document outlines the bare-bones, minimal code for a basic NodeMCU program, as well as a program for sending & receiving messages to & from an MQTT broker. All the code samples can be downloaded from this repository:

https://github.com/coppercloud-iotech/iot-foundation-wksp

1. Basic "Blink" Program Structure



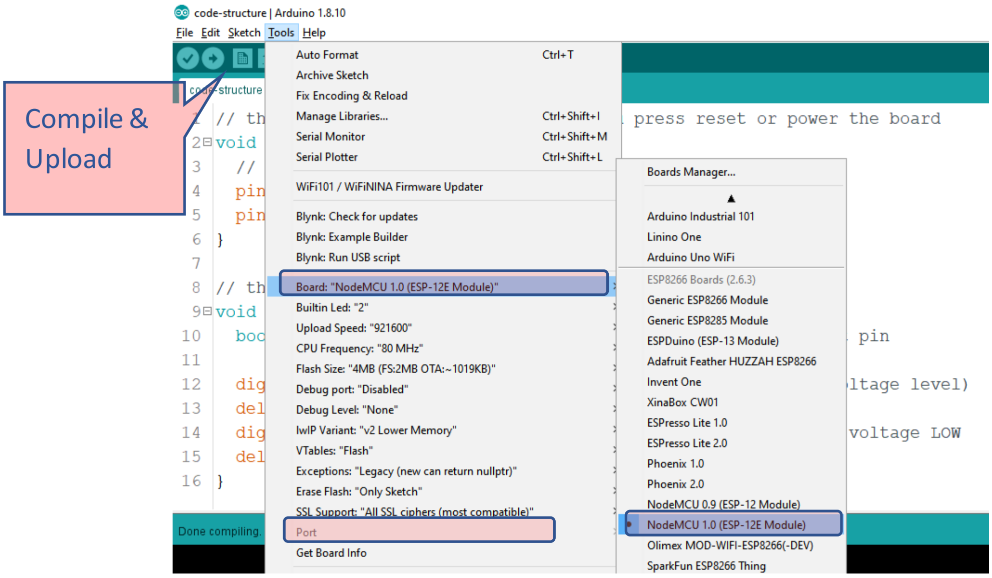
```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(13, OUTPUT);
  pinMode(D3, INPUT);
}

// the loop function runs over and over again forever
void loop() {
  boolean pinState = digitalRead(D4);    // read state of input pin

  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

This function is run only ONCE, when the MCU starts. Put all initialization/PIN setup code here.

This function is run repeatedly, non-stop, until the MCU loses power. Put all the repeating code (such as reading sensors, responding to messages, etc here)

Read Input PIN State

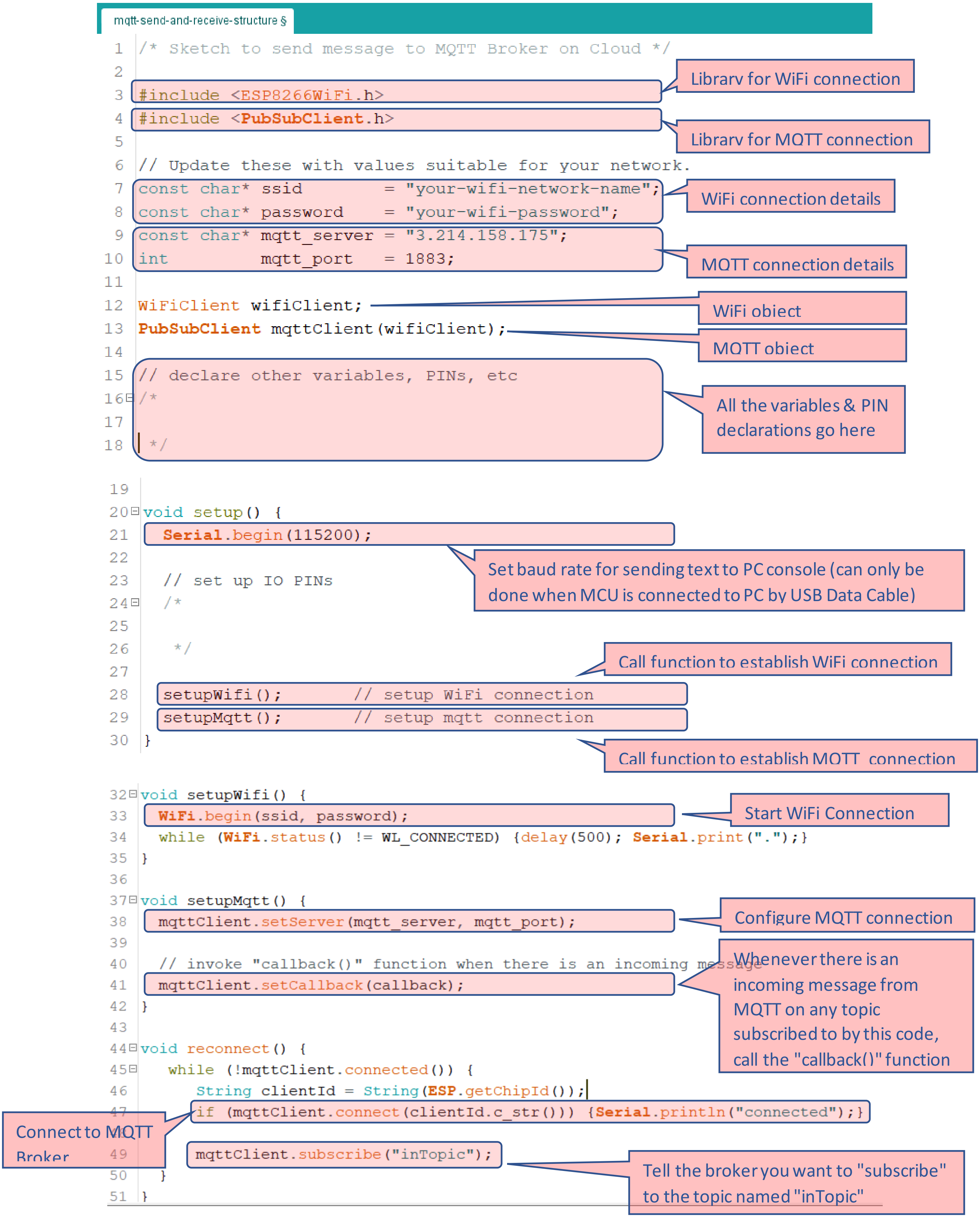
Change Output PIN State



Board: "NodeMCU 1.0 (ESP-12E Module)"
Builtin Led: "2"
Upload Speed: "921600"
CPU Frequency: "80 MHz"
Flash Size: "4MB (FS:2MB OTA:~1019KB)"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
VTables: "Flash"
Exceptions: "Legacy (new can return nullptr)"
Erase Flash: "Only Sketch"
SSL Support: "All SSL ciphers (most compatible)"

Compile & Upload

2. Program Structure to Send/Receive Messages to/from MQTT Broker



```
/* Sketch to send message to MQTT Broker on Cloud */

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

// Update these with values suitable for your network.
const char* ssid          = "your-wifi-network-name";
const char* password      = "your-wifi-password";
const char* mqtt_server = "3.214.158.175";
int           mqtt_port    = 1883;

WiFiClient wifiClient;
PubSubClient mqttClient(wifiClient);

// declare other variables, PINs, etc
/*

 */

void setup() {
  Serial.begin(115200);

  // set up IO PINs
  /*

   */

  setupWifi();         // setup WiFi connection
  setupMqtt();         // setup mqtt connection
}

void setupWifi() {
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {delay(500); Serial.print(".");}
}

void setupMqtt() {
  mqttClient.setServer(mqtt_server, mqtt_port);

  // invoke "callback()" function when there is an incoming message
  mqttClient.setCallback(callback);
}

void reconnect() {
   while (!mqttClient.connected()) {
      String clientId = String(ESP.getChipId());
      if (mqttClient.connect(clientId.c_str())) {Serial.println("connected");}

      mqttClient.subscribe("inTopic");
   }
}
```

Library for WiFi connection

Library for MQTT connection

WiFi connection details

MQTT connection details

WiFi object

MQTT object

All the variables & PIN declarations go here

Set baud rate for sending text to PC console (can only be done when MCU is connected to PC by USB Data Cable)

Call function to establish WiFi connection

Call function to establish MQTT connection

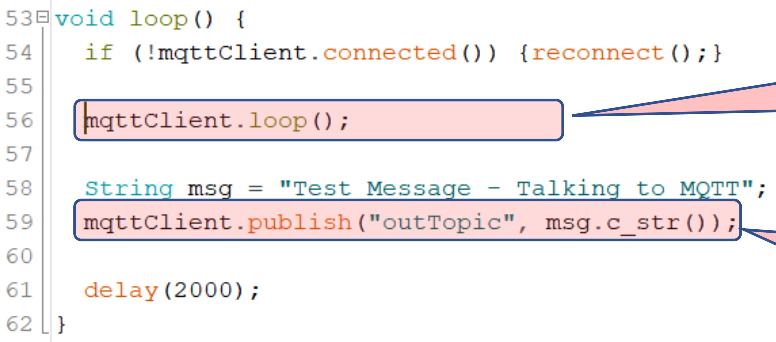
Start WiFi Connection

Configure MQTT connection

Whenever there is an incoming message from MQTT on any topic subscribed to by this code, call the "callback()" function

Connect to MQTT Broker

Tell the broker you want to "subscribe" to the topic named "inTopic"



```
void loop() {
  if (!mqttClient.connected()) {reconnect();}

  mqttClient.loop();

  String msg = "Test Message - Talking to MQTT";
  mqttClient.publish("outTopic", msg.c_str());

  delay(2000);
}
```

This statement makes sure NodeMCU keeps listening for incoming messages in every loop

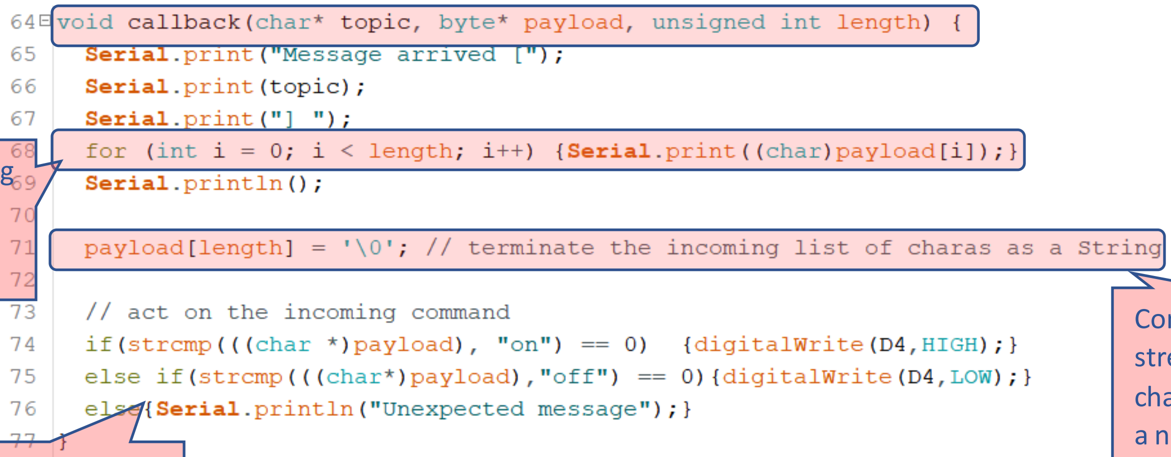
This is how we "publish" messages to a specific Topic on the MQTT Broker ("outTopic" in this case)

Whenever there is an incoming message on any of your subscribed Topics, "loop()" is interrupted/halted, and this function is called.

"topic" = which of your subscribed Topics has received the message

"payload" = contents of the message

"length" = length of the messages (how many bytes)



```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {Serial.print((char)payload[i]);}
  Serial.println();

  payload[length] = '\0'; // terminate the incoming list of charas as a String

  // act on the incoming command
  if(strcmp(((char *)payload), "on") == 0)  {digitalWrite(D4,HIGH);}
  else if(strcmp(((char*)payload),"off") == 0){digitalWrite(D4,LOW);}
  else{Serial.println("Unexpected message");}
}
```

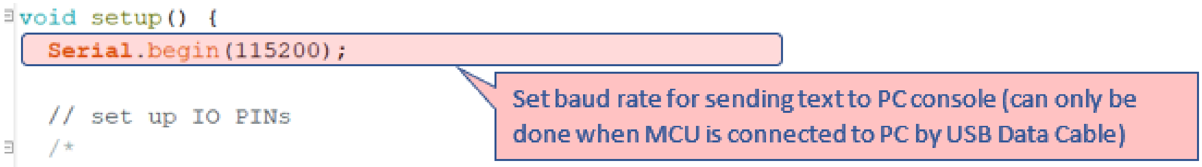
Print incoming message on console

Convert your stream of characters into a null-terminated string (required for C/C++)

Compare incoming message to set of expected values, and take required action (such as reading a sensor, or activating an output PIN

3. How to check text output on Arduino IDE Console

We put this line of code in the setup() function, to start sending text to the Arduino IDE console (for debugging or monitoring output of program:



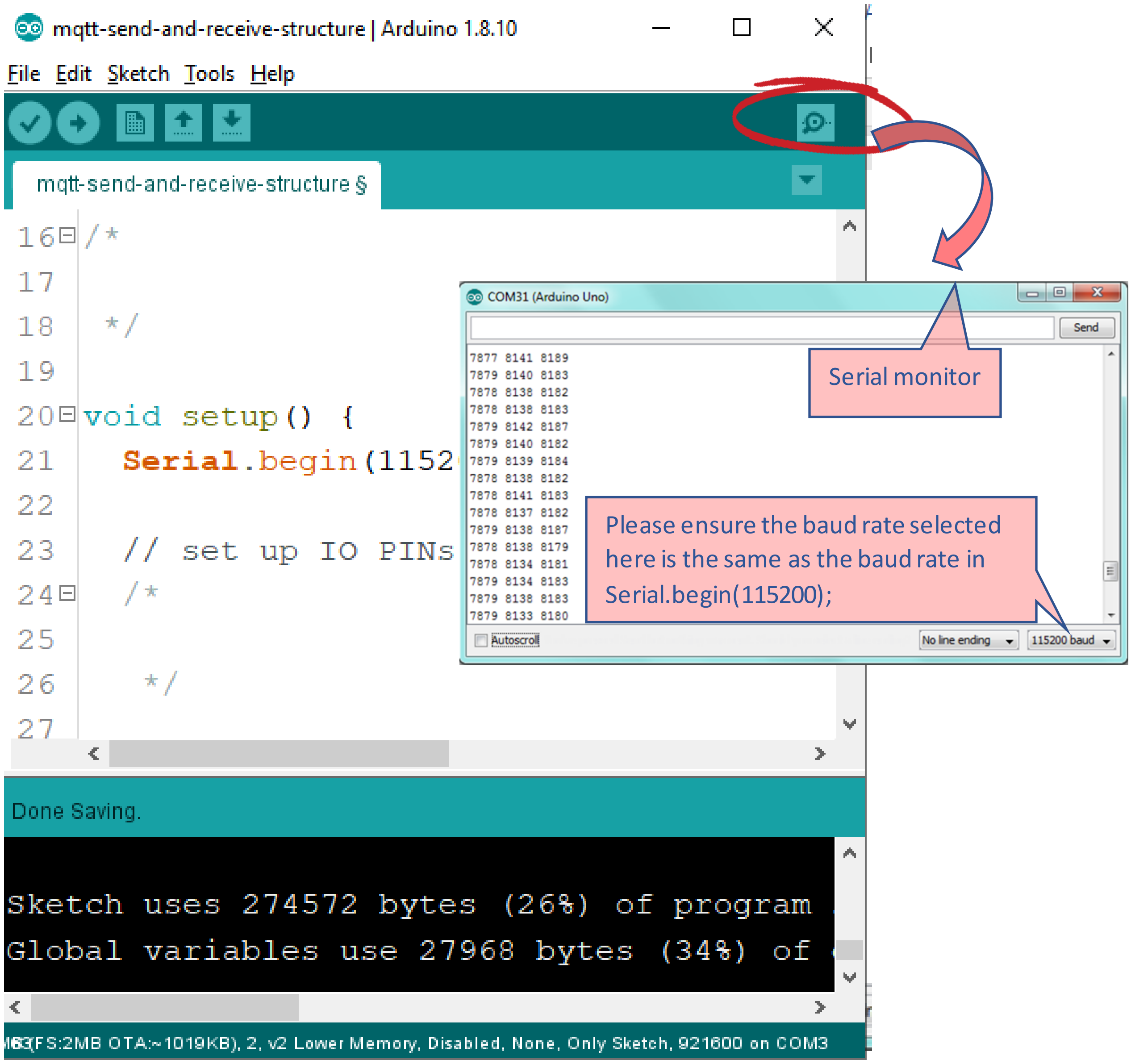
```
void setup() {
  Serial.begin(115200);

  // set up IO PINs
  /*
```

Set baud rate for sending text to PC console (can only be done when MCU is connected to PC by USB Data Cable)

And we send text to the console by these (and similar) lines of code:

```
Serial.println("connected");
```

To monitor this text output on the console/serial monitor, follow these steps:



mqtt-send-and-receive-structure | Arduino 1.8.10
File Edit Sketch Tools Help
mqtt-send-and-receive-structure §

```
/*

 */

void setup() {
  Serial.begin(1152

  // set up IO PINs
  /*

   */

```

COM31 (Arduino Uno)

```
7877 8141 8189
7879 8140 8183
7878 8138 8182
7878 8138 8183
7879 8142 8187
7879 8140 8182
7879 8139 8184
7878 8138 8182
7878 8141 8183
7878 8137 8182
7879 8138 8187
7878 8138 8179
7878 8134 8181
7879 8134 8183
7879 8138 8183
7879 8133 8180
```

□ Autoscroll No line ending 115200 baud

Serial monitor

Please ensure the baud rate selected here is the same as the baud rate in Serial.begin(115200);

Done Saving.

```
Sketch uses 274572 bytes (26%) of program
Global variables use 27968 bytes (34%) of
```

M3(FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 921600 on COM3