# Lab Report

Using a motor with a joystick | Controlling a relay over a web server

## Goal of the experiment

Two experiments are performed.

### Using a motor with a joystick

In the first experiment, a *stepper motor* is used with a *joystick* equipped with a switch. Input from the joystick on the Y-axis determines how fast the motor rotates, and whether the rotation is performed clockwise or counterclockwise. When pressed, the switch updates the current mode: manual – listening for input from the joystick – or "autopilot" – performing revolutions a certain number of times.

### Controlling a relay over a web server

In the second experiment, a *relay* is controlled over a web server hosted on an *ESP8266* to allow switching an *external LED* on or off.

## Components

### Using a motor with a joystick

- 1x Arduino UNO
- 1x **B103 3810** joystick
- 1x **28BYJ-48** stepper motor with a **ULN2003** driver board
- 4x 1.5V batteries (in a holder case)
- 1x Bobby pin (attached to the stepper motor)
- Jumper wires

### Controlling a relay over a web server

- 1x **ESP8266**
- 1x micro-USB to USB-A cable
- 1x Arduino UNO
- 1x **JQC-3FF-S-Z** relay
- 1x LED
- 1x 220 Ohm resistor
- Jumper wires

## Relevant theory

### Using a motor with a joystick

A stepper motor can rotate continuously and be positioned precisely – similarly to servo motors – relying on a cogged wheel and electromagnets to nudge the wheel round one step at a time (Last Minute Engineers, n.d.). The cogged wheel is rotated towards either of the electromagnets receiving a high pulse, one at a time. Rotating the wheel is achieved when the phases are energized one after the other in the right sequence, known as a *step sequence* (Last Minute Engineers, n.d.).

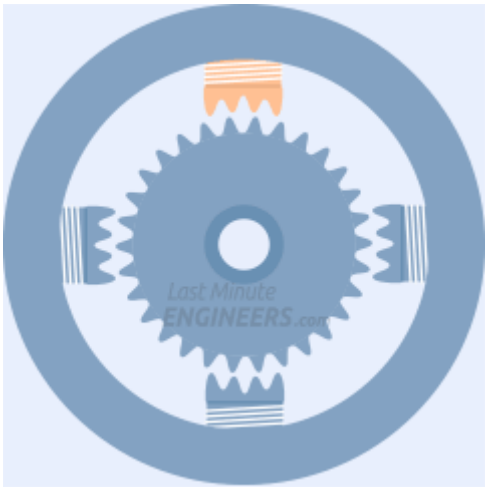



*Figure 1 - Rotation of the cog wheel with electromagnets*

Several adjustments can be made to operate the motor in different ways, depending on the way the coils are pulsed. What property is changed and its effect on the wheel is illustrated in the table below (Last Minute Engineers, n.d.).

| Property changed about the pulses | Property of the motor being affected |
|---|---|
| **Sequence** | Spinning direction |
| **Frequency** | Speed |
| **Amount** | How far the motor will turn |

The 28BYJ-48 has specifications worth becoming familiar with. Some of those are listed in the table below.

| Description | Value |
|---|---|
| **Powered drawn** | around 240mA |
| **Operating voltage (current type)** | 5V (DC) |
| **Number of phases** | 4 |
| **Number of coils** | 2 |


*Figure 2 - 28BYJ-48 stepper motor*

Control over the 28BYJ-48 stepper motor is made possible through the ULN2003 driver board, able to deal with the motor's high power consumption – something not possible with the Arduino. The figure below displays the driver's components, among which are control inputs, the ULN2003 integrated circuit (IC), the motor connection, power inputs, on and off jumpers as well as 4 step indicator LEDs (Last Minute Engineers, n.d.).

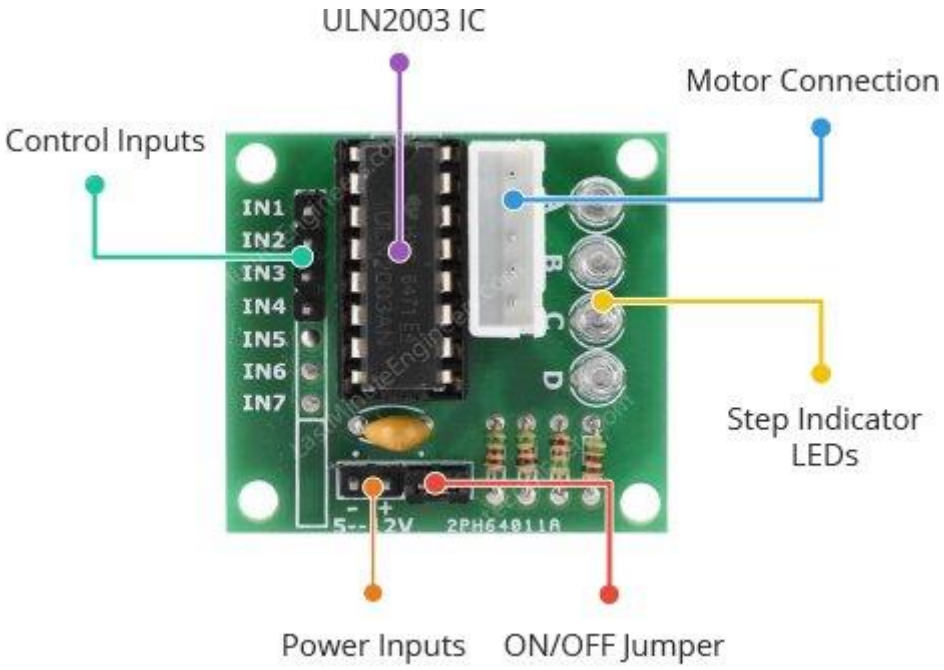



*Figure 3 - ULN2003 driver board*

The B103 3810 joystick is equipped with two potentiometers, each working on a specific axis (horizontal X-axis or vertical Y-axis), with a switch working similarly to a button. When reading analog values from either of the potentiometers, the values will range from 0 to 1023.

### Bibliography

Last Minute Engineers. (n.d.). *Control 28BYJ-48 Stepper Motor with ULN2003 Driver & Arduino*. https://lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial/

## Controlling a relay over a web server

Despite all its possible applications, the Arduino cannot control high-voltage devices, as it runs on 5 volts. To be able to achieve this using the Arduino, a possible solution involves using a *relay* module (Last Minute Engineers, n.d.).

A relay is a component composed of an electromagnet acting as an electric lever: a small current through the relay allows a large current to flow. When the small current stops flowing, so does the large one. This is caused by the small current creating a magnetic field around the electromagnet of the relay, attracting contact with the other circuit where the large current can now flow (Last Minute Engineers, n.d.). An illustration of this process is shown below.

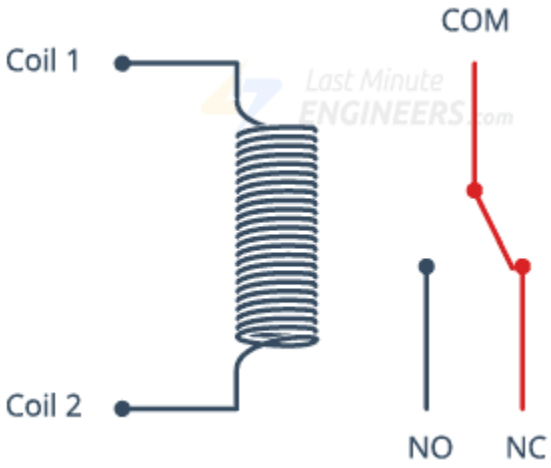


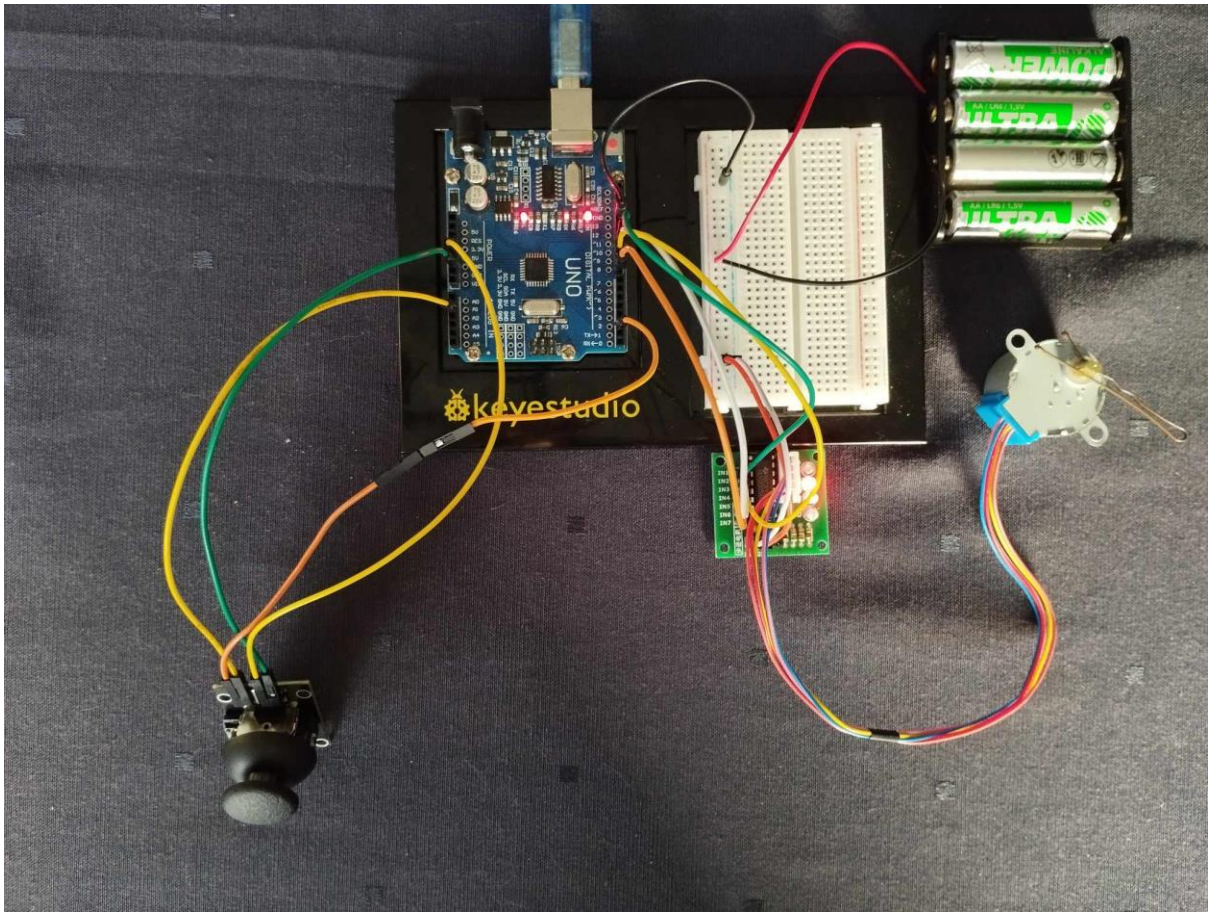

*Figure 4 - Relay module internal representation*

## Bibliography

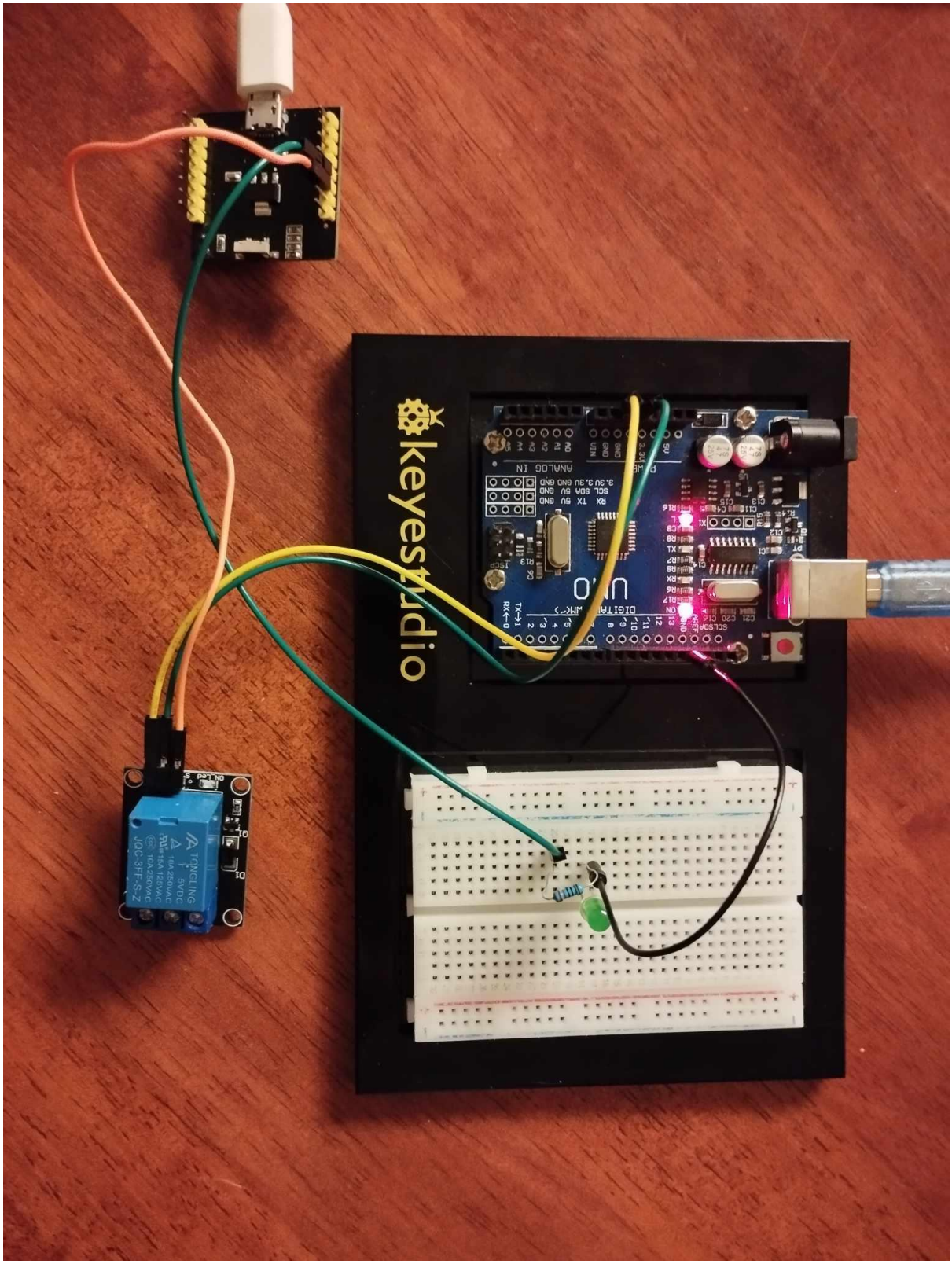
Last Minute Engineers. (n.d.). *Interface One Channel Relay Module with Arduino*. https://lastminuteengineers.com/one-channel-relay-module-arduino-tutorial

# Setup description

## Using a motor with a joystick




## Controlling a relay over a web server




## Source code

### Using a motor with a joystick

```
#include <Stepper.h>

#define Y_PIN   A1
```

```
#define SW_PIN  2
#define SW_INITIAL_STATE  HIGH
#define Y_AT_REST  500

#define IN1 11
#define IN2 10
#define IN3 9
#define IN4 8
#define DEFAULT_STEPS         0
#define STEPS_PER_REVOLUTION  2038
#define DEFAULT_SPEED         5
#define MAX_SPEED             15
#define DEFAULT_AUTOPILOT     true
#define AUTOPILOT_REVOLUTIONS 1

bool autopilot = false;
int revolutions_counter = 0;
int steps, motor_speed, clockwise;

bool last_released_state = (SW_INITIAL_STATE == HIGH) ? true : false;
Stepper stepper = Stepper(STEPS_PER_REVOLUTION, IN1, IN3, IN2, IN4);

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

  // Joystick
  pinMode(SW_PIN, INPUT);
  digitalWrite(SW_PIN, HIGH);
  Serial.println("Joystick ready for use.");

  // Stepper
  set_motor_values(autopilot);
  Serial.println("Stepper motor ready for use.");

  delay(3500);
}

void loop() {
  // Handle SWITCH button click
  bool current_released_state = digitalRead(SW_PIN);
  if (last_released_state != current_released_state) {
    String state = current_released_state ? "RELEASED" : "PRESSED";
    Serial.println("!\t" + state + "\t!");
    if (state == "PRESSED") {
      switch_mode();
    }
  }
  last_released_state = current_released_state;
```

```
  if (!autopilot) {
    // Read Y value from joystick
    int y = analogRead(Y_PIN);

    if (y != Y_AT_REST) {
      // Adjust speed, direction and steps to non-null Y-axis value
      clockwise = (y > Y_AT_REST);
      motor_speed = map(y, 0, 1023, 0, MAX_SPEED);
      steps = STEPS_PER_REVOLUTION / MAX_SPEED;
      if (!clockwise) steps *= -1;
    } else {
      steps = 0;
    }
  }

  Serial.println("Mode\t-\t" + String(autopilot ? "AUTOPILOT" : "MANUAL"));
  Serial.println("Speed\t-\t" + String(motor_speed));
  Serial.println("Steps\t-\t" + String(steps));
  if (autopilot) Serial.println("Revolution #\t-\t" +
String(revolutions_counter) + " / " + String(AUTOPILOT_REVOLUTIONS));
  Serial.println("==================================");

  if (!autopilot || revolutions_counter < AUTOPILOT_REVOLUTIONS) {
    if (motor_speed > 0) stepper.setSpeed(motor_speed);
    if (steps != 0) stepper.step(steps);
    if (autopilot) revolutions_counter++;
  }

  delay(250);
}

void switch_mode() {
  autopilot = !autopilot;
  set_motor_values(autopilot);
}

void set_motor_values(bool autopilot) {
  if (autopilot) {
    clockwise = true;
    motor_speed = MAX_SPEED;
    steps = STEPS_PER_REVOLUTION;
    revolutions_counter = 0;
  } else {
    motor_speed = DEFAULT_SPEED;
    steps = STEPS_PER_REVOLUTION / MAX_SPEED;
  }
}
```

## Controlling a relay over a web server

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

#define LED_PIN 4
#define RELAY_PIN 5

#ifndef STASSID
#define STASSID "<SSID>"
#define STAPSK "<PASSWORD>"
#endif

const char* ssid = STASSID;
const char* password = STAPSK;

String s;  // Web page content
ESP8266WebServer server(80);

void handleRoot() {
  server.send(200, "text/html", s);
}

void handleNotFound() {
  digitalWrite(LED_PIN, 1);
  String message = "File Not Found\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET) ? "GET" : "POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i = 0; i < server.args(); i++) { message += " " +
server.argName(i) + ": " + server.arg(i) + "\n"; }
  server.send(404, "text/plain", message);
  digitalWrite(LED_PIN, 0);
}

void handleLedOn() {
  digitalWrite(LED_PIN, 1);
  redirect_home();
}

void handleLedOff() {
  digitalWrite(LED_PIN, 0);
  redirect_home();
}
```

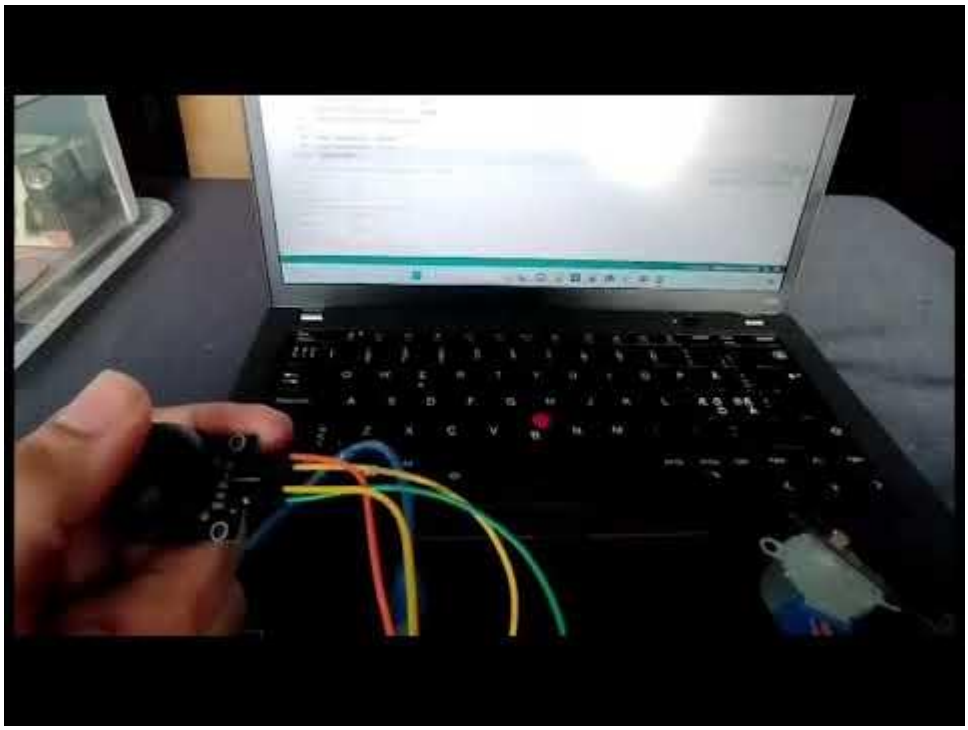
```
void handleRelayOn() {
  digitalWrite(RELAY_PIN, 1);
  redirect_home();
}

void handleRelayOff() {
  digitalWrite(RELAY_PIN, 0);
  redirect_home();
}

void redirect_home() {
  server.sendHeader("Location", "/");
  server.send(303);
}

void setup(void) {
  // Start serial
  Serial.begin(115200);

  // Prepare GPIO4 (LED)
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, 0);

  // Prepare GPIO5 (relay)
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, 0);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  s = "<!DOCTYPE html>";
  s += "<html>";
  s += "<head>";
  s += "<meta name='viewport' content='width=device-width, initial-scale=1'>";
  s += "<link
href='https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css'
```

```
rel='stylesheet' integrity='sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN'
crossorigin='anonymou'>";
  s += "</head>";
  s += "<body>";
  s += "<div class='container'>";
  s += "<h1>LED</h1>";
  s += "<div class='row g-2 mb-3'>";
  s += "<div class='col-12 col-md-6 col-lg-4'>";
  s += "<a class='btn btn-primary w-100' href='/LedOn'>ON</a>";
  s += "</div>";
  s += "<div class='col-12 col-md-6 col-lg-4'>";
  s += "<a class='btn btn-danger w-100' href='/LedOff'>OFF</a>";
  s += "</div>";
  s += "</div>";
  s += "<h1>Relay</h1>";
  s += "<div class='row g-2'>";
  s += "<div class='col-12 col-md-6 col-lg-4'>";
  s += "<a class='btn btn-primary w-100' href='/RelayOn'>ON</a>";
  s += "</div>";
  s += "<div class='col-12 col-md-6 col-lg-4'>";
  s += "<a class='btn btn-danger w-100' href='/RelayOff'>OFF</a>";
  s += "</div>";
  s += "</div>";
  s += "</div>";
  s += "</body>";
  s += "</html>";

  server.on("/", handleRoot);
  server.onNotFound(handleNotFound);

  // Handle turning the LED on / off
  server.on("/LedOn", handleLedOn);
  server.on("/LedOff", handleLedOff);

  // Handle turning the relay on / off
  server.on("/RelayOn", handleRelayOn);
  server.on("/RelayOff", handleRelayOff);

  server.begin();
  Serial.println("HTTP server started");
}

void loop(void) {
  server.handleClient();
}
```

# Result

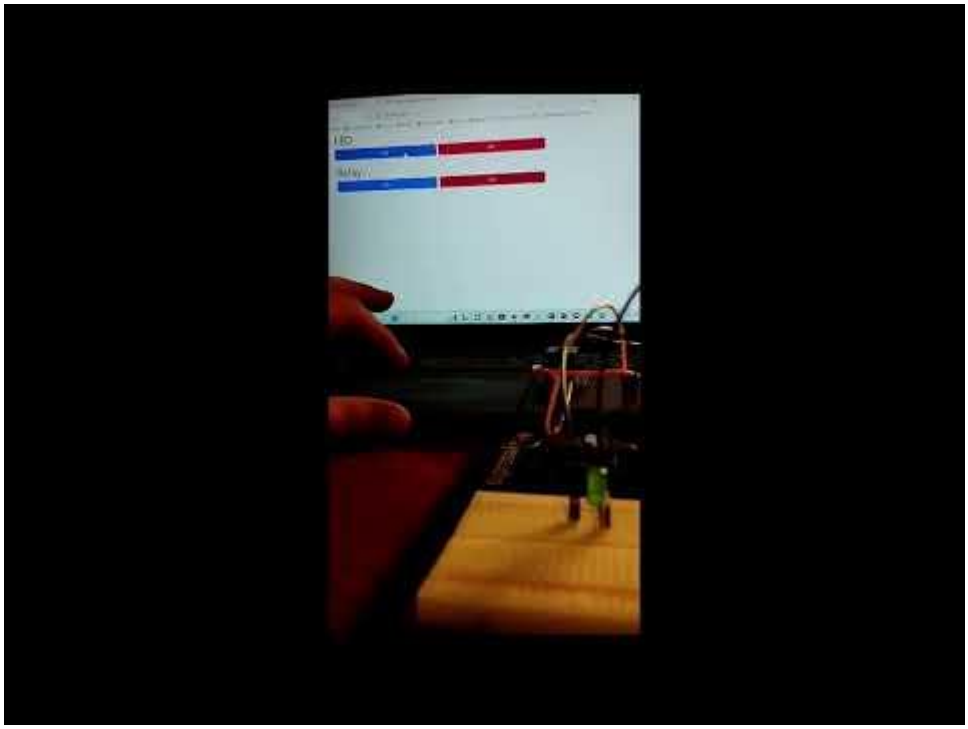
## Using a motor with a joystick

Expected results have been achieved for this experiment, allowing the stepper motor to be controlled using the Y-axis of the joystick. Moving the joystick upwards along this axis will rotate the motor clockwise, while moving downwards will rotate it counterclockwise. In addition, the speed is increased the more the joystick is pushed to its boundaries on the Y-axis. Finally, two modes can be alternated when pressing on the switch: in the manual mode, the speed and orientation can be controlled directly, while the “autopilot” mode will perform a specific number of revolutions before stopping the motor.



## Controlling a relay over a web server

Expected results have been achieved for this experiment as well. Both the relay and LED can be turned on and off independently and over the same network the ESP8266 is connected to.



# Reflection

## Using a motor with a joystick

Motor are essential components proving their use in all possible fields that require or supports level of automation. With a stepper motor, small and precise movements can be performed, allowing for good control and reliability over the movement being performed. Adding a joystick to control the speed and orientation has created an extra challenge that proved efficient in learning to use a stepper motor, also in creative ways.

## Controlling a relay over a web server

Web servers can be used for hosting web pages that interact with their environments. This experiment supports other applications to web servers than their traditional web hosting. In scenarios similar to the one shown in this experiment, components could be coupled with the relay to be activated given certain requirements whose state can be changed directly over the web.