runlinc Intermediate Project I9: Robot Movement Control (E32W Version)

Contents

Introduction

Problem

Robots have gradually become a significantly important part in people's lives and work.

How can we control the basic movements of robots?

Background

A robot is a machine—especially one programmable by a computer—capable of carrying out a complex series of actions automatically. A robot can be guided by an external control device, or the control may be embedded within. Robots may be constructed to evoke human form, but most robots are task-performing machines, designed with an emphasis on stark functionality, rather than expressive aesthetics.

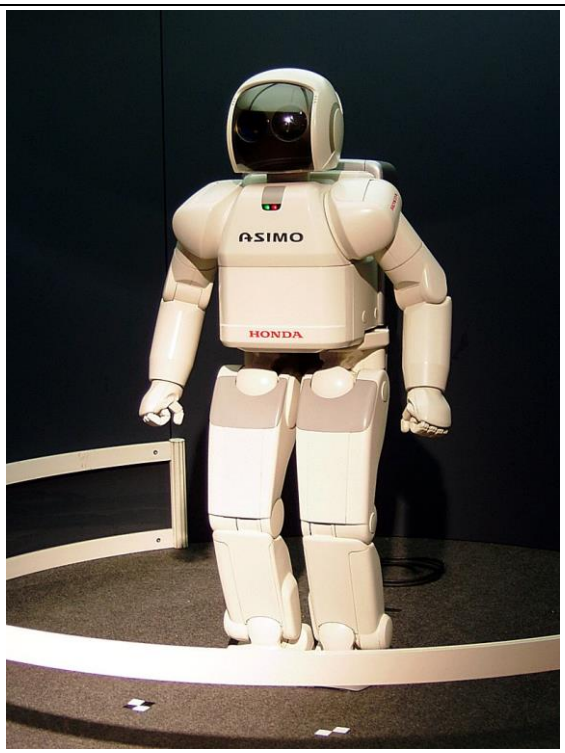

Figure 1: ASIMO is a humanoid robot created by Honda in 2000.

One of the most common appliances of robots can be seen in our daily life is the robotic vacuums cleaner.

Figure 2: Household robotic vacuum cleaner

Robotic vacuum cleaners don't use cameras to see the world. Instead, they use various types of sensors to detect and measure the worlds around them and their own progress through it, including cliff sensors, bump sensors, wall sensors and optical encoders. Cliff sensors measure the distance between the robot base and the floor, usually by bouncing infra-red light off the floor. If there is a sudden

increase in the distance to the floor, that means the robot is getting close to a stair edge or something similar, so it will back off to avoid falling over it (hence the "cliff sensor" name).

The name of the bump sensors also gives away what they do: if the robot vacuum bumps into something (like a wall or a chair leg), the impact triggers the sensor.

Wall sensors are like cliff sensors, but in a different direction: they tell the robot when it is close to a wall or other object, so it can follow the wall.

Optical encoders are the most important: these sensors on the wheels of the robot tell it how far it has gone. They are called optical encoders because they use a light sensor to detect how many times the wheels have rotated. From this (and any difference between wheels, which indicates a turn), the robot can figure out how far it has traveled. Different models may include additional sensors (such as a dust scanner to see how much dust is being picked up), but those are the basic sensors that all robotic vacuums include.

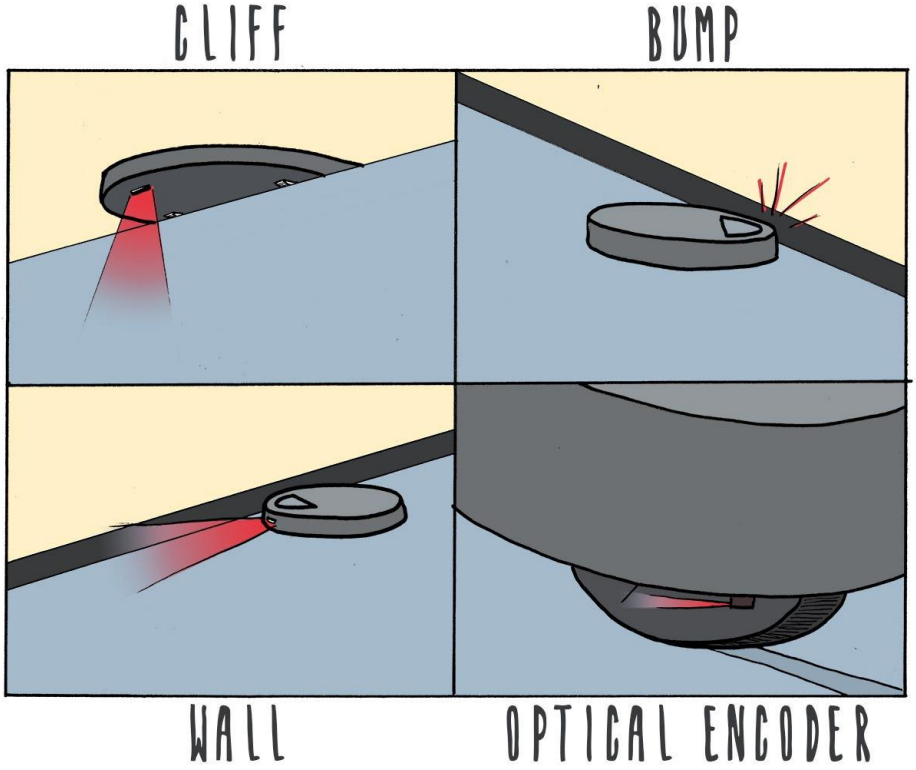

Figure 3: Application of sensors in robotic vacuum cleaner

Ideas

How does the robot move? How to control the robot to turn? How can we use a website to simulate basic movements of a robot? How can we make the robot automatically control its movements?

Plan

We will use the DC motors, wheels, and robot base to simulate the basic movement of robots. We will create five buttons to control the movements of robots.

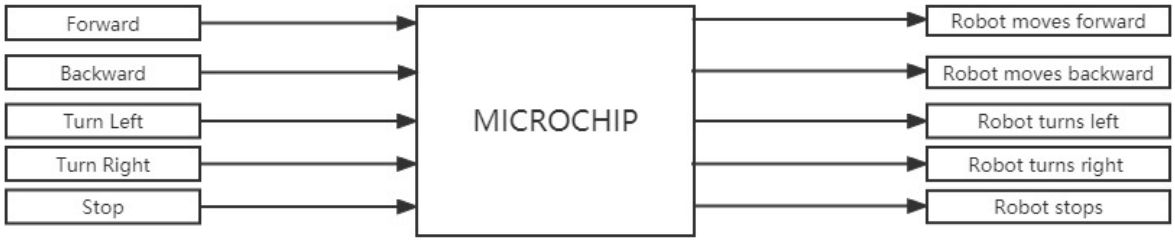

Figure 4: Block diagram of Microchip inputs and outputs

runlinc Background

runlinc is a web page inside a Wi-Fi chip. The programming is done inside the browsers compare to programming inside a chip. The runlinc web page inside the Wi-Fi chip will command the microchips to do sensing, control, data logging Internet of Things (IoT). It can predict and command.

Part A: Design the Circuit on runlinc

Note: Refer to runlinc Wi-Fi Setup Guide document to connect to runlinc
Use the left side of the runlinc web page to construct an input/output (I/O).

For port D12 name it right_forward and set it as DIGITAL_OUT.

For port D13 name it right_back and set it as DIGITAL_OUT.

For port D14 name it enable and set it as DIGITAL_OUT.

For port D27 name it left_forward and set it as DIGITAL_OUT.

For port D33 name it left_back and set it as DIGITAL_OUT.

Figure 5: I/O configurations connections

Part B: Build the Circuit

Use the STEMSEL E32W board to connect the hardware. For this project we are using the screw terminals(GND, VIN, D13, D12, D27 and D33) on the bottom of the E32W board.

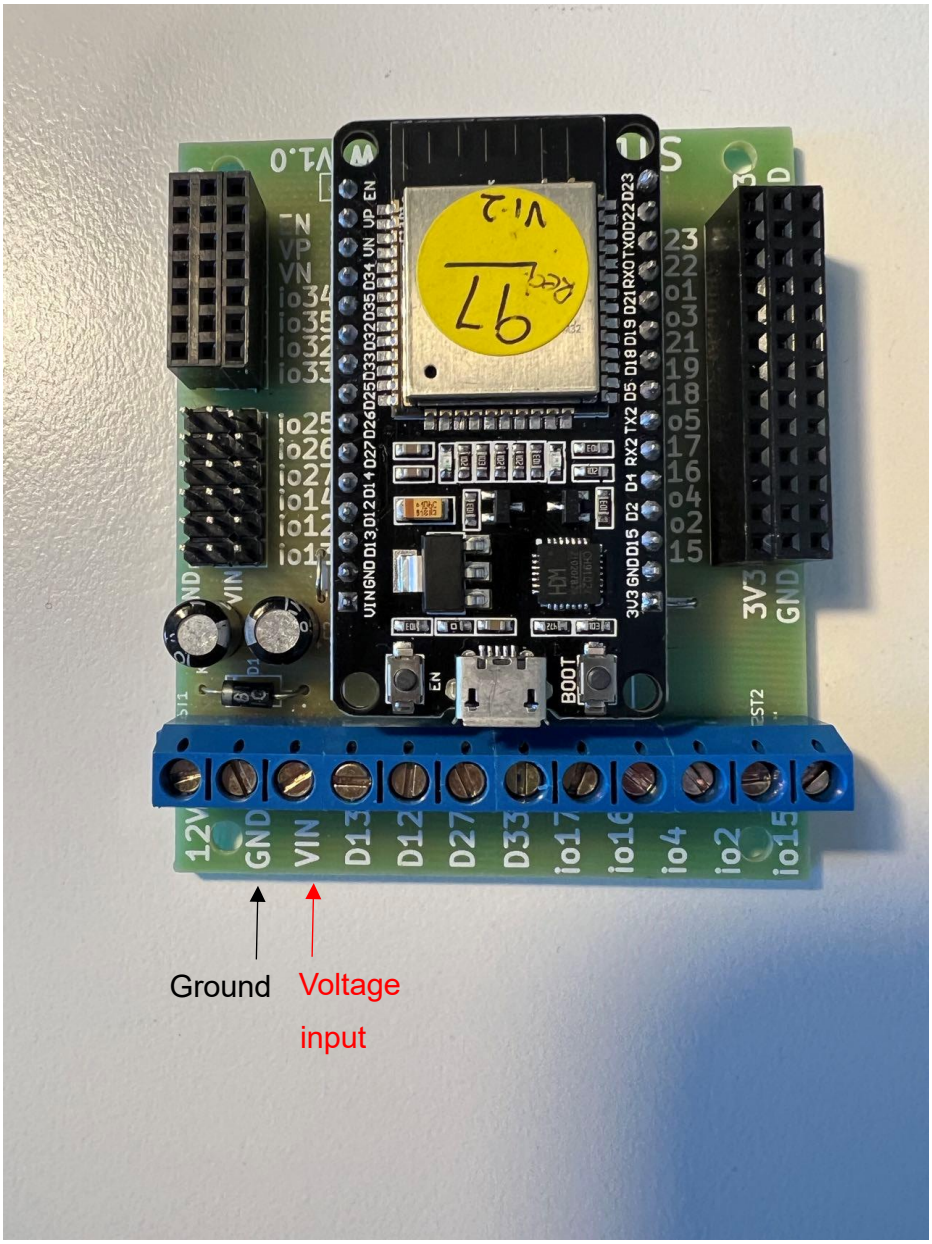

Figure 6: Negative, Positive and Signal port on the E32W board

There is a robot base with two DC Motors, two big black wheels, a small white wheel and a battery holder box shown in the figure below. Each motor has two wires (red and black) that control the direction of rotation of the wheel. And two wires of the battery holder box can be used to supply power to the E32W board.

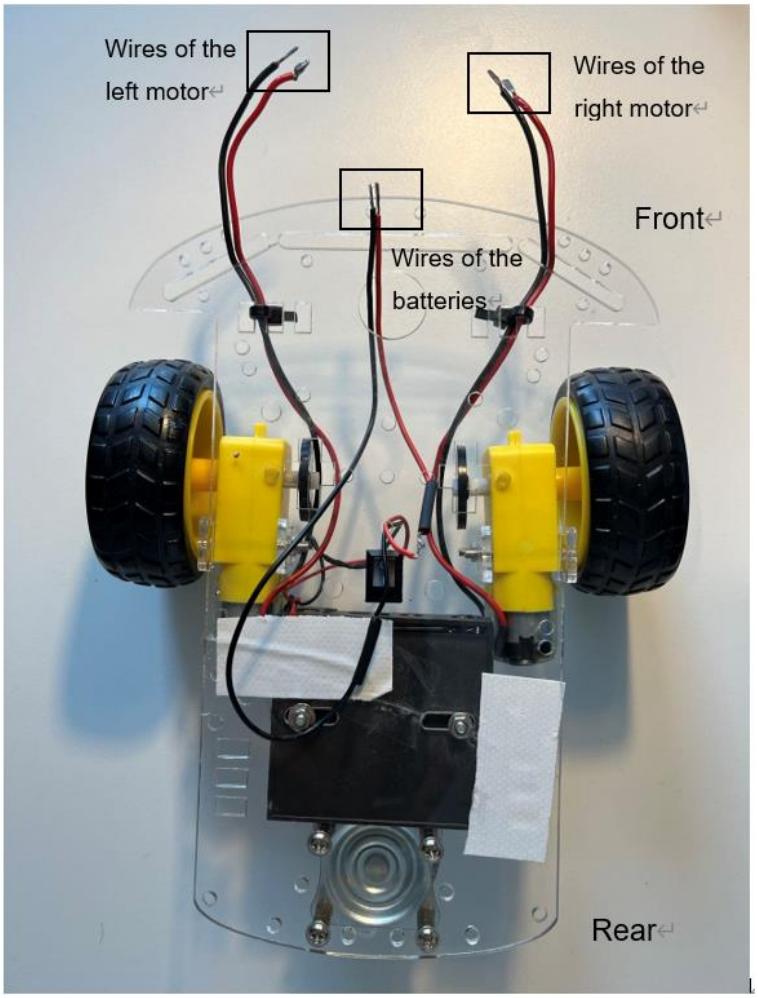

Figure 7: The positive side of the robot

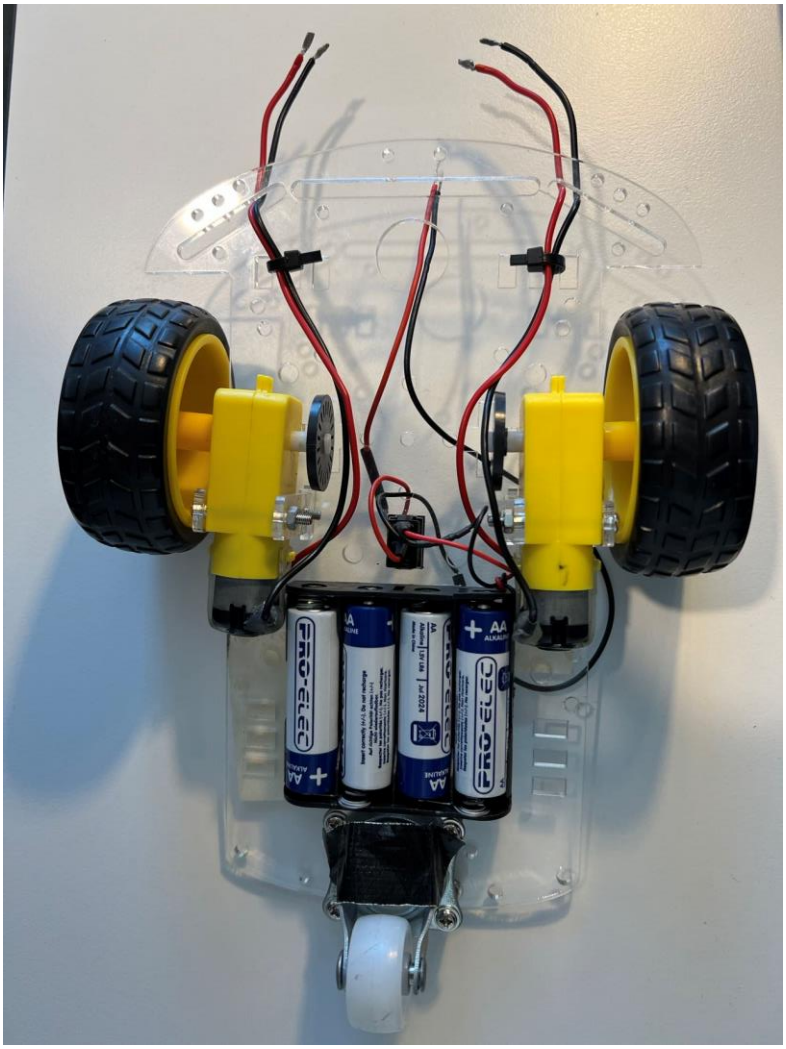

Figure 8: The reverse side of the robot

Wiring instructions

a.) Equip four batteries in the battery holder box.

b.) Unscrew GND and VIN pins, then connect black wire of the battery holder box to the GND pin and red wire of the battery box to the VIN pin.

c.) Unscrew D33 and D27 pins, then connect black wire of the left motor to the D27 pin and red wire to the D33 pin and.

d.) Unscrew D12 and D13 pins, then connect black wire of the right motor to the D12 pin and red wire to the D13 pin and. Don't forget to tighten the screws.

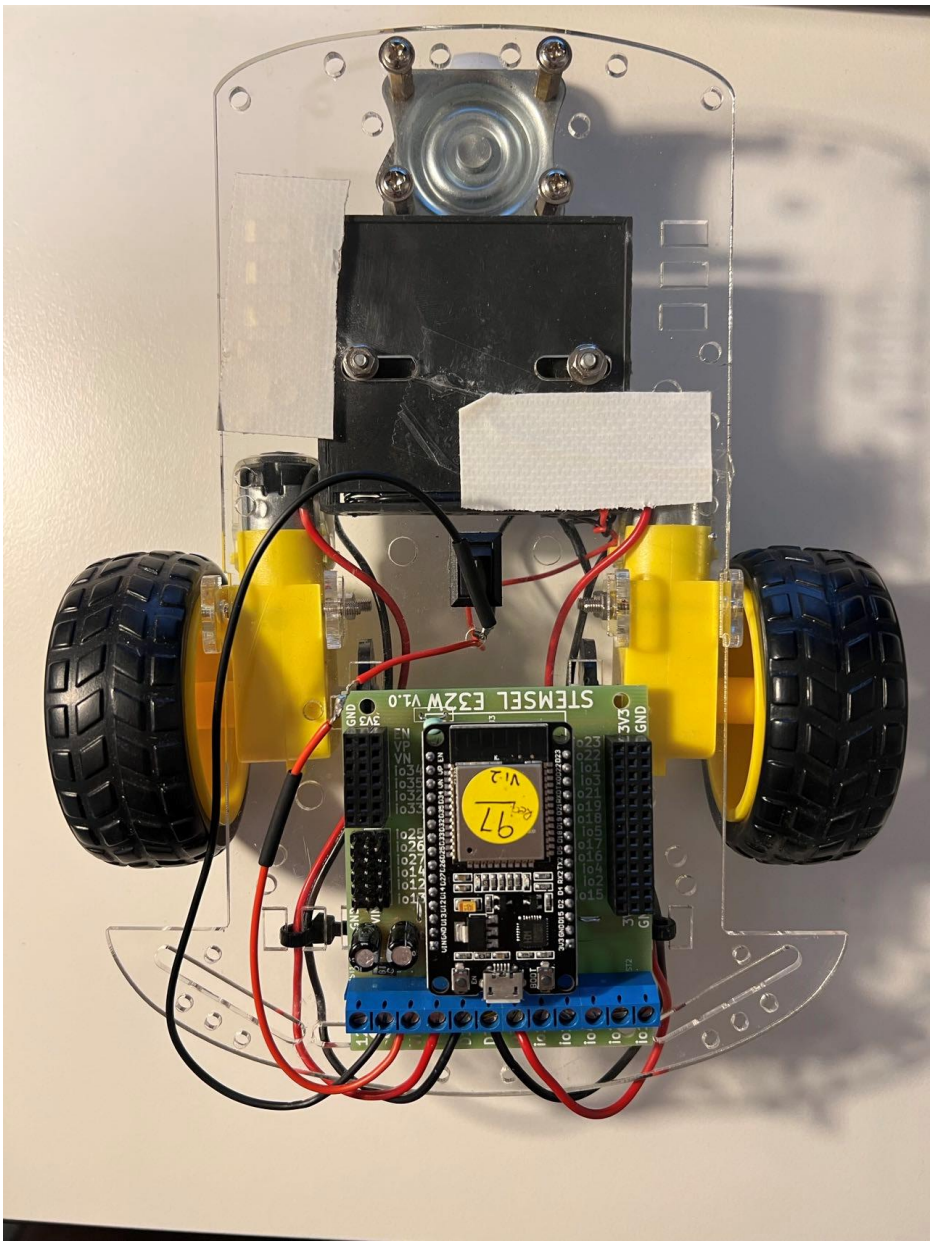

Figure 9: Circuit board connection with screw terminal part (top view)

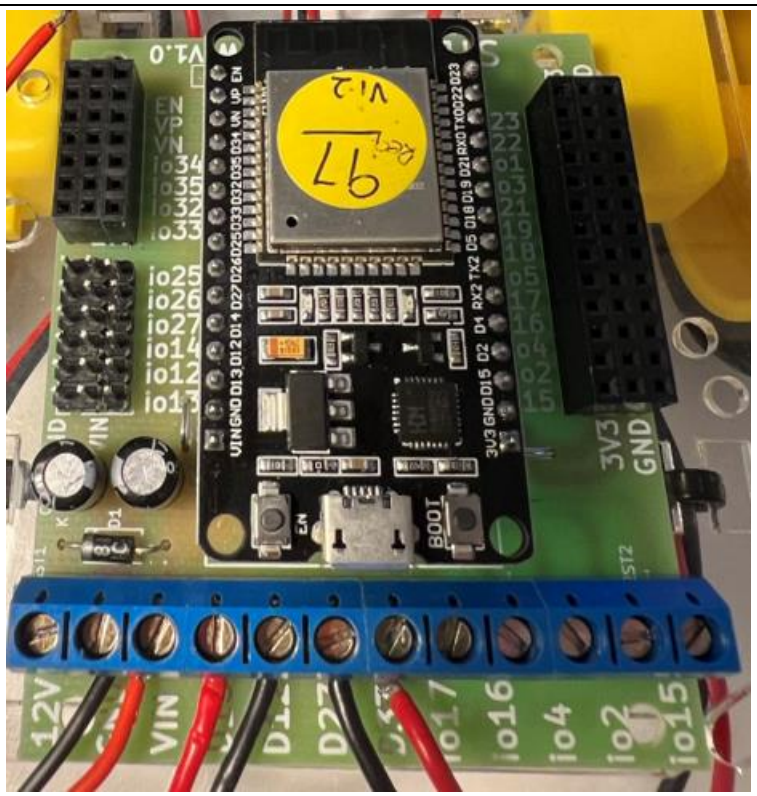

Figure 10: Circuit board connection with screw terminal part (zoom in)

Part C: Program the Circuit

CSS:

In the CSS block, we are going to want to move to make sure the button is in the right spot for what we want:

```
button{
 margin: 0.5rem;
 }
```

HTML:

1. Under HTML, we will be displaying the title of the webpage, Robot Control Webpage.

```
<h2> Robot Control Webpage </h2>
```

2. After creating the title, we will start to design five buttons. Firstly, we will create the button named ‘Forward’. The function of this button is to keep robot moving forward.

```
<button onclick="forward()"> Forward </button>
```

3. Then, the second button named ‘Backward’ is to keep robot moving backward.

```
<button onclick="backward()"> Backward </button>
```

4. And then, we need to generate two buttons named ‘Turn Left’ and ‘Turn Right’ to turn the robot left and right respectively.

```
<button onclick="turnleft()"> Turn Left </button>
<button onclick="turnright()"> Turn Right </button>
```

5. Lastly, we need to create a button which stop the robot which means turn off all the other buttons.

```
<button onclick="stop()"> Stop </button>
```


JavaScript:

We previously named D14 to enable but there isn't any I/O part to connect, this is due to D14 on runlinc page is used to enable the driver output (D12, D13, D27 and D33 screw terminals). The screw terminals are set as 'disabled' as the driver chip uses quite a bit of power, even it's not driving anything, therefore turning it off to save power (especially useful with the power source is batteries). Firstly, we need to enable D14 using the turnOn Macro, we will program this feature in Part C in the JavaScript box. If somehow your E32W board works without turning on D14, please also include this line of code.

```
turnOn( enable );
```

Then, we need to create the first function 'forward' which enable the left and right wheels rotate counterclockwise. And this function can force the robot moving forward.

```
function forward(){
turnOn( left_forward );
turnOn( right_forward );
}
```

And then, we need to create the second function 'backward' which enable the left and right wheels rotate clockwise. And this function can force the robot moving backward.

```
function backward(){
turnOn( left_back );
turnOn( right_back );
}
```

Now, we need to create the third and fourth function 'turn left' and 'turn right' which enable the robot turn left and right.

```
function turnleft(){
turnOn( right_forward );
}
function turnright(){
turnOn( left_forward );
}
```

Lastly, we need to create the fifth function 'stop' which turn off all the other buttons and it stops the robot.

```
function stop(){
turnOff( left_forward );
turnOff( right_forward );
turnOff( left_back );
turnOff( right_back );  }
```


Final Code:

The final code for CSS block:

```
button{
 margin: 0.5rem;
 }
```

The final code for HTML block:

```
<h2> Robot Control Webpage </h2>
<button onclick="forward()"> Forward </button>
<button onclick="backward()"> Backward </button>
<br>
<br>
<button onclick="turnleft()"> Turn Left </button>
<button onclick="turnright()"> Turn Right </button>
<br>
<br>
<button onclick="stop()"> Stop </button>
```

The final code for JavaScript block:

```
turnOn( enable );
function forward(){
turnOn( left_forward );
turnOn( right_forward );
}
function backward(){
turnOn( left_back );
turnOn( right_back );
}
function turnleft(){
turnOn( right_forward );
}
function turnright(){
turnOn( left_forward );
}
function stop(){
turnOff( left_forward );
turnOff( right_forward );
turnOff( left_back );
turnOff( right_back );
}
```


Expected Result

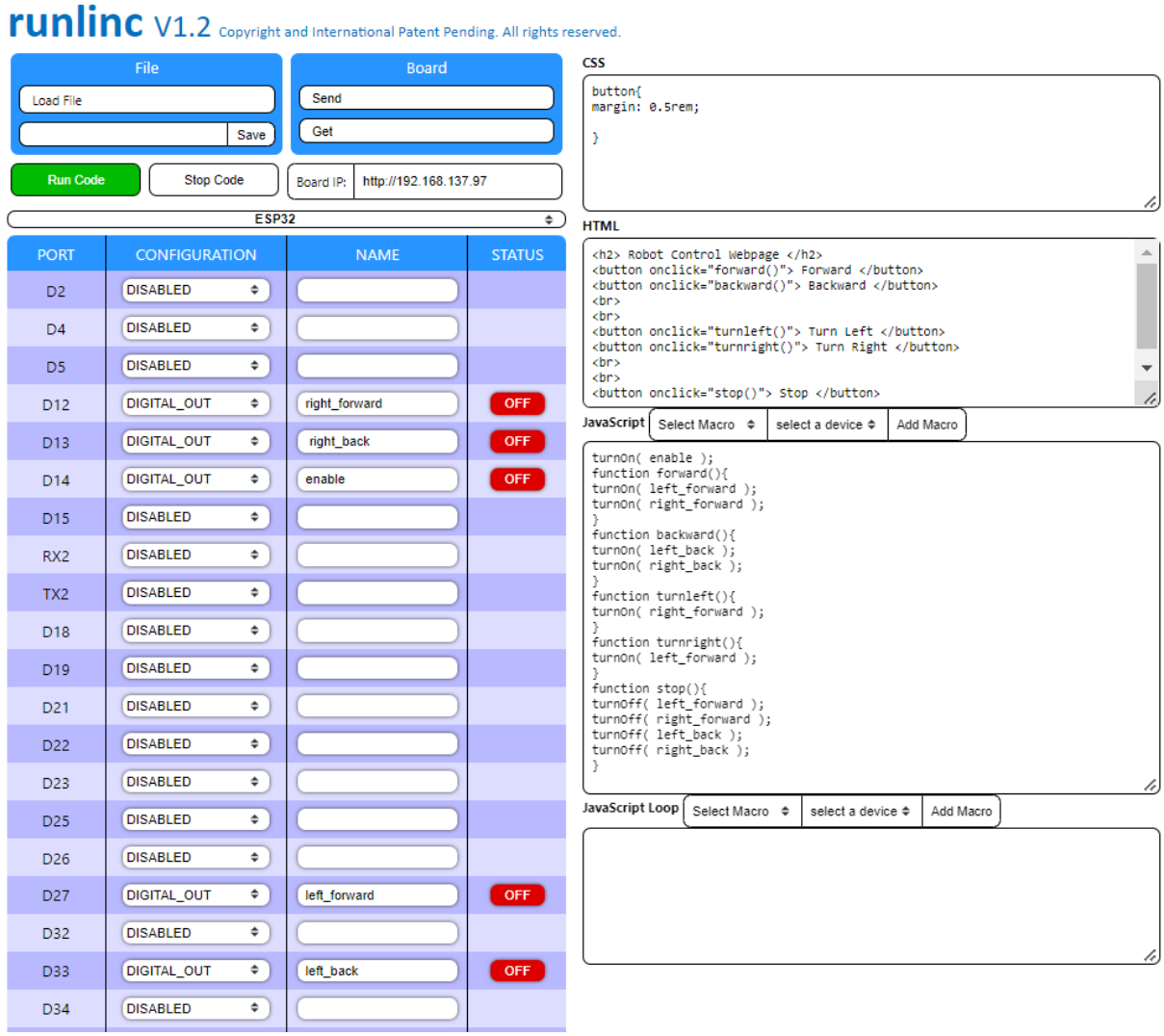

Figure 11: Expect runlinc result screenshot

Challenge

What if we want to control the robot to respond automatically using a Light Dependent Resistor (LDR) rather than clicking buttons manually.

Part A: Design the Circuit on runlinc

Note: Refer to runlinc Wi-Fi Setup Guide document to connect to runlinc
Use the left side of the runlinc web page to construct an input/output (I/O).

For port D12 name it right_forward and set it as DIGITAL_OUT.

For port D13 name it right_back and set it as DIGITAL_OUT.

For port D14 name it enable and set it as DIGITAL_OUT.

For port D27 name it left_forward and set it as ANALOG_IN.

For port D32 name it lightsensor and set it as DIGITAL_OUT.

For port D33 name it left_back and set it as DIGITAL_OUT.

D12	DIGITAL_OUT	right_forward	OFF
D13	DIGITAL_OUT	right_back	OFF
D14	DIGITAL_OUT	enable	OFF
D15	DISABLED		
RX2	DISABLED		
TX2	DISABLED		
D18	DISABLED		
D19	DISABLED		
D21	DISABLED		
D22	DISABLED		
D23	DISABLED		
D25	DISABLED		
D26	DISABLED		
D27	DIGITAL_OUT	left_forward	OFF
D32	ANALOG_IN	lightsensor	19
D33	DIGITAL_OUT	left_back	OFF

Figure 12: I/O configurations connections

Part B: Build the Circuit

Based on the E32W board equipped in the previous steps which is shown in the figure below

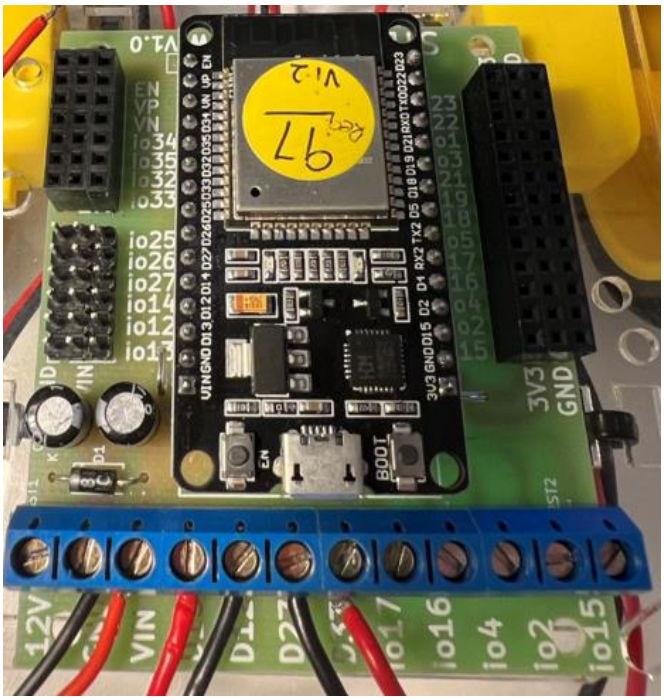

Figure 13: Circuit board connection with screw terminals (zoom in)

We need a Light Dependent Resistor (LDR) module (KY-018) and their respective pins are shown in the figure below.

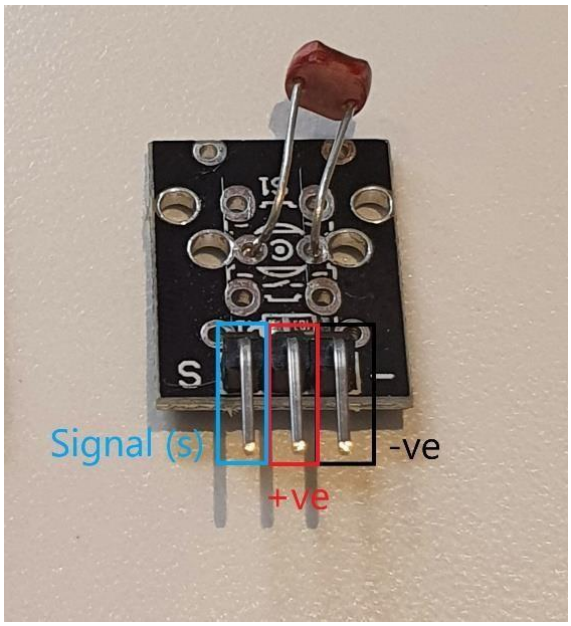

Figure 14: I/O part with negative, positive and signal pins indicated

Wiring instructions

a.) Plug in the Light Dependent Resistor (LDR) to io32 on the E32W board.

b.) Make sure the (-ve) pin are on the GND (outer) side of the I/O ports.

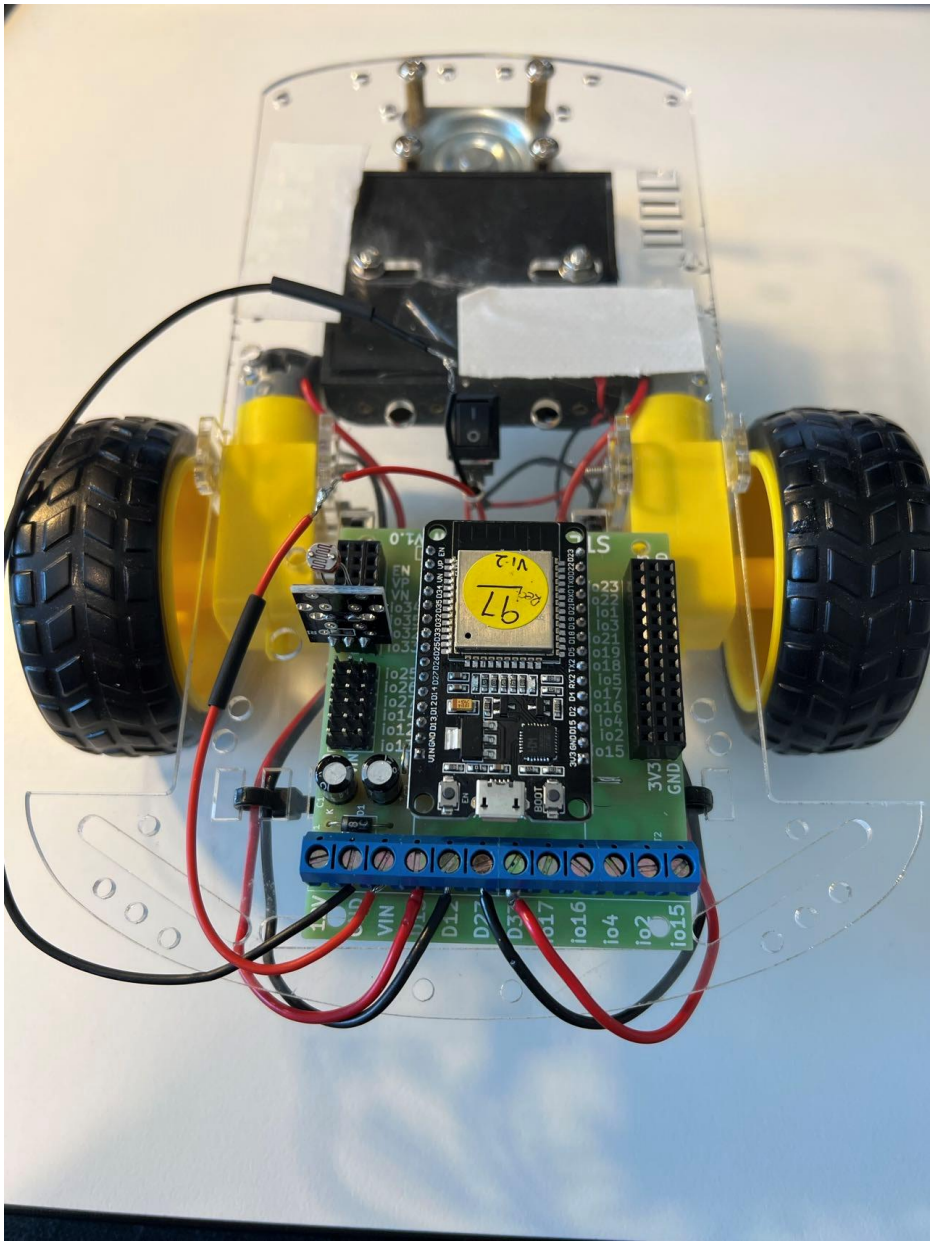

Figure 15: Circuit board connection with screw terminals and I/O part (top view)

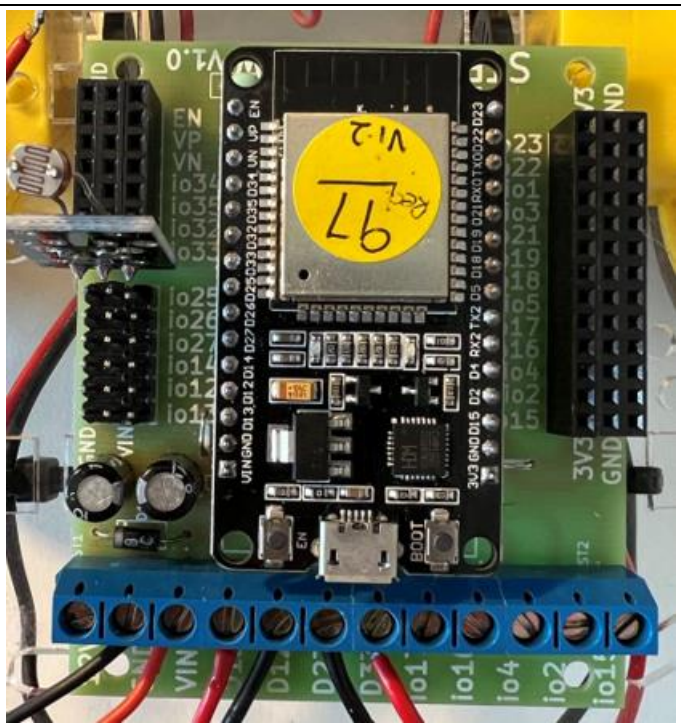

Figure 16: Circuit board connection with I/O part (zoom in)

Part C: Program the Circuit

CSS:

In the CSS block, we are going to want to move to make sure the button is in the right spot for what we want and enlarge the buttons a little bit:

```
button{
 margin: 0.5rem;
padding: 12px 28px;
 }
```

HTML:

Now we only need two buttons ‘Start’ and ‘Brake’ in the HTML section. We don’t need any other buttons created before.

```
<h2> Robot Control Webpage </h2>
<button onclick="start()">Start </button>
<button onclick="brake()">Brake</button>
```

JavaScript:

For JavaScript section type the following code:

```
function start(){
turnOn( enable );
}
function brake(){
turnOff( enable );
}
function forward(){
turnOn( left_forward );
turnOn( right_forward );
}
function backward(){
turnOn( left_back );
turnOn( right_back );
}
function turnleft(){
turnOn( right_forward );
}
```



```
function turnright(){
turnOn( left_forward );
}
function stop(){
turnOff( left_forward );
turnOff( right_forward );
turnOff( left_back );
turnOff( right_back );
}
```

JavaScript Loop:

For JavaScript Loop section type the following code:

```
if(analogIn( lightsensor )>140 ){
stop();
backward();
}else if(analogIn( lightsensor )>80){
stop();
turnright();
}else{ stop();
forward();
}
```

Then an ‘if else statement loop’ is added to the Light Dependent Resistor (LDR) for it to decide the movement of the robot. If the reading is above 140, the robot moves backward. However, if the reading is below 140 and above 80, the robot turns right. Furthermore, if the reading is below 80, the robot keeps moving forward. Remember our Light Dependent Resistor (LDR) have a higher reading when its surrounding is darker, and a lower reading when its surrounding is brighter.

Final Code:

The final code for CSS block:

```
button{
 margin: 0.5rem;
padding: 12px 28px;
 }
```

The final code for HTML block:

```
<h2> Robot Control Webpage </h2>
<button onclick="start()">Start </button>
<button onclick="brake()">Brake</button>
```

The final code for JavaScript block:

```
function start(){
turnOn( enable );
}
function brake(){
turnOff( enable );
}
function forward(){
turnOn( left_forward );
turnOn( right_forward );
}
function backward(){
turnOn( left_back );
turnOn( right_back );
}
function turnleft(){
turnOn( right_forward );
}
function turnright(){
turnOn( left_forward );
}
function stop(){
turnOff( left_forward );
turnOff( right_forward );
```



```
turnOff( left_back );
turnOff( right_back );
}
```

The final code for JavaScript Loop block:

```
if(analogIn( lightsensor )>140 ){
stop();
backward();
}else if(analogIn( lightsensor )>80){
stop();
turnright();
}else{ stop();
forward();
}
```


Expected Result

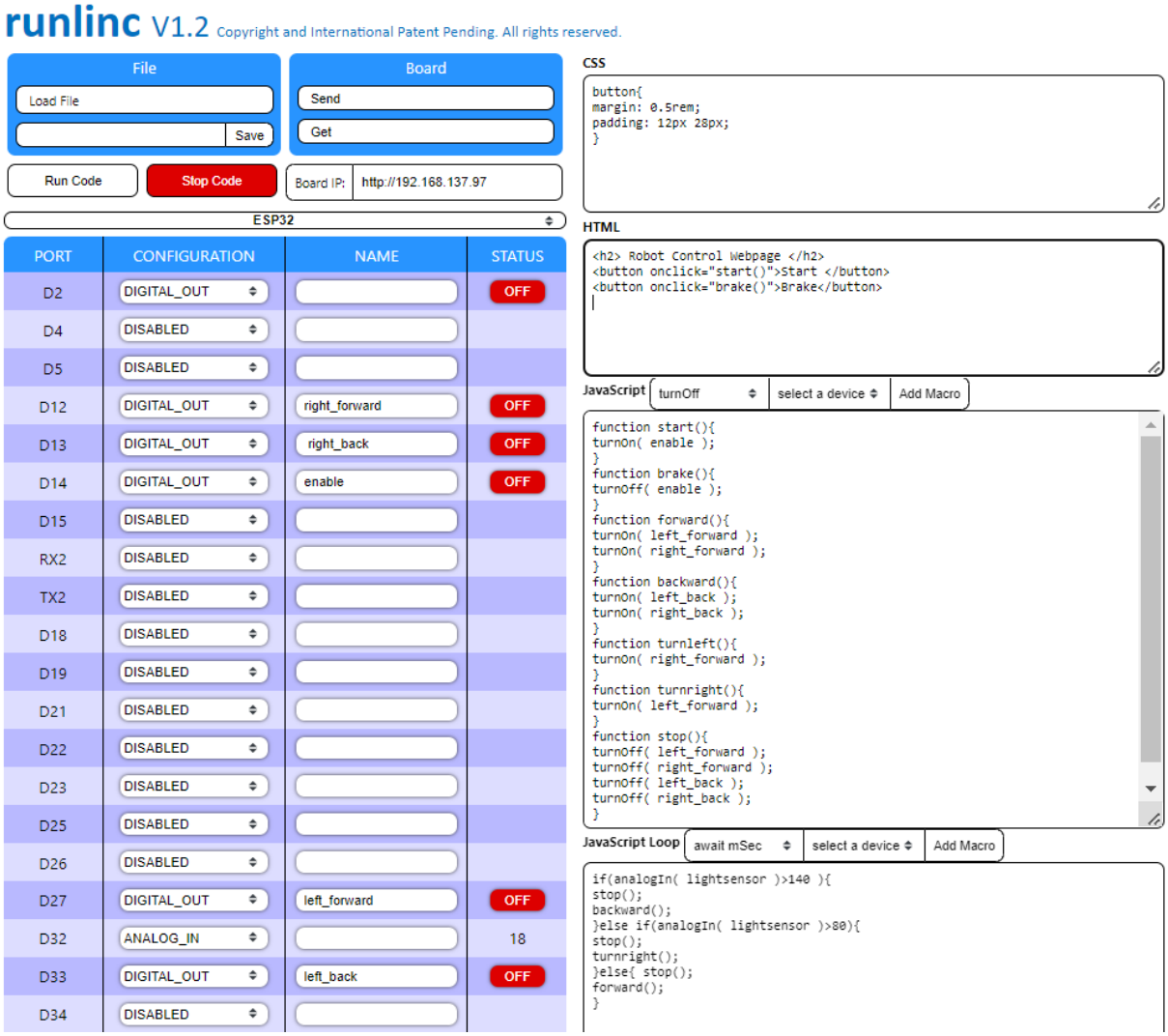

Figure 17: Expect runlinc result screenshot

Summary

Robot is playing a more and more important role in people's daily life. In this project, we have learnt how to program and create buttons to control the basic movements of the robot.

