

**MICROCONTROLLERS FOR MECHATRONICS – MECA442**

Experiment 7: soft pwm

**Jana Kabrit1, Oussama Al Ahmad Matar1**

**1** Rafik Hariri University, College of Engineering, Department of Mechanical and Mechatronics

*Abstract*

*The aim of this experiment is to design and simulate a circuit that would* *generate a PWM signal out of a digital pin with a frequency of 500Hz. Two pushbuttons will be used to control the speed and direction of the motor by controlling the duty cycle of the PWM signal.*

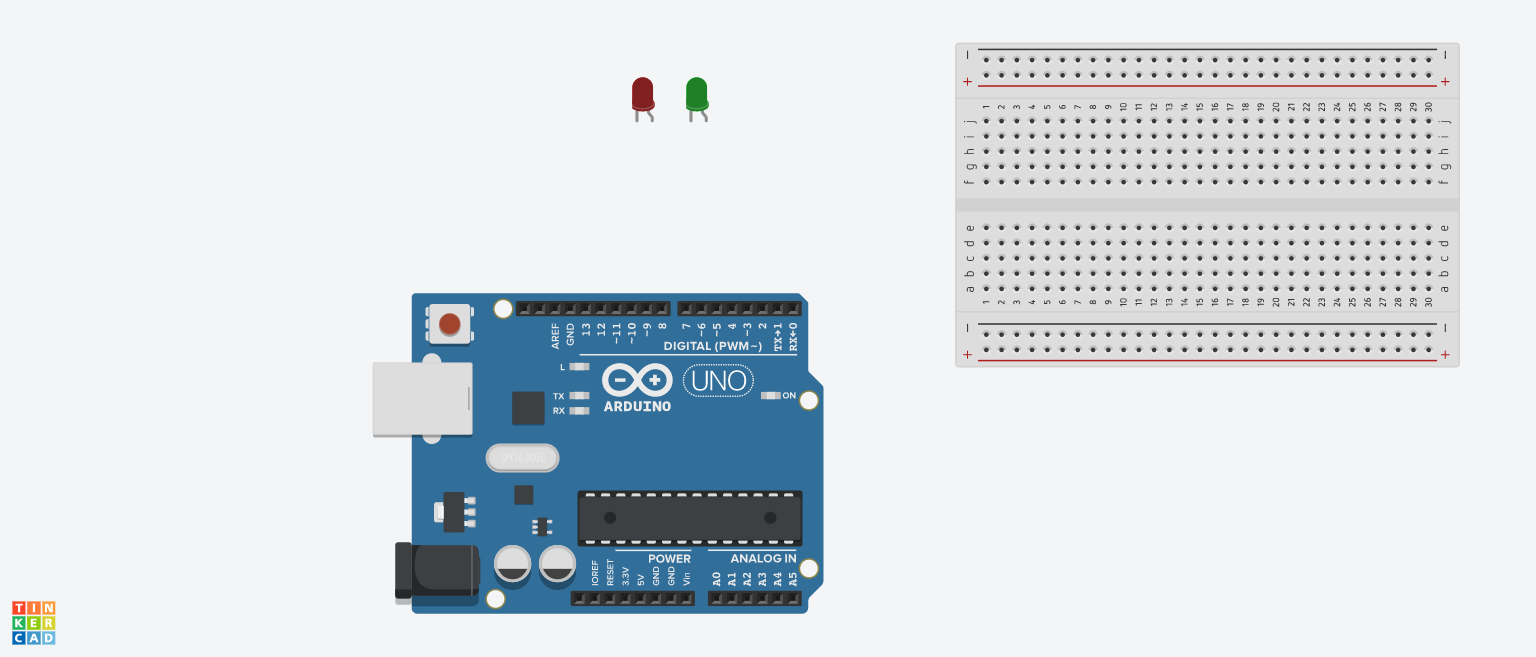
**Keywords:** Arduino Uno, Soft PWM, Duty Cycle, TinkerCAD.

1. INTRODUCTION

In this experiment, a soft PWM signal will be generated from a digital pin of the Arduino Uno, where PWM, or Pulse Width Modulation, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. The PWM signal will have a frequency of 500Hz (or a total period of 2 milliseconds per pulse). A pushbutton will be used to control the speed of the motor by increasing the duration of the duty cycle of each pulse, which is the active or HIGH state, while maintaining the same PWM frequency. A second pushbutton will be used to control the direction of rotation of the motor.

1. **MATERIALS AND METHODS**
   1. **Materials**
      1. **Simulated Electronics Components**

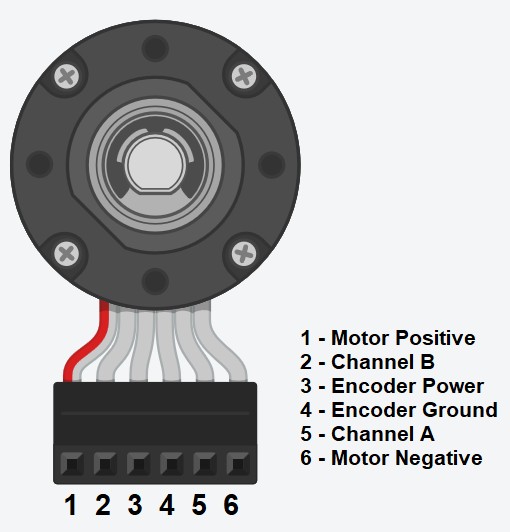
* Arduino Uno

The Arduino Uno (Figure 1) is a microcontroller board based on the ATmega328. It has 20 digital input/output pins (of which 6 can be used as PWM outputs and 6 can be used as analog inputs), a 16 MHz resonator, a USB connection, a power jack, an in-circuit system programming (ICSP) header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.[1] In this experiment, an Arduino is used to create a soft PWM signal from a digital pin to mimic the properties of an analog pin and vary the motor speed.

**FIGURE 1:** ARDUINO UNO BOARD

* DC Motor:

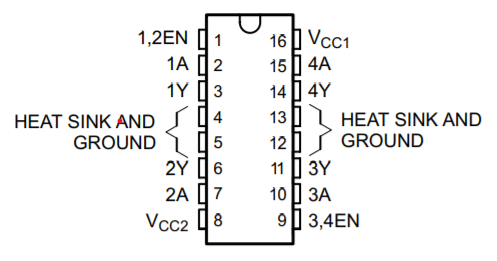
**FIGURE 2:** DC MOTOR WITH ENCODER



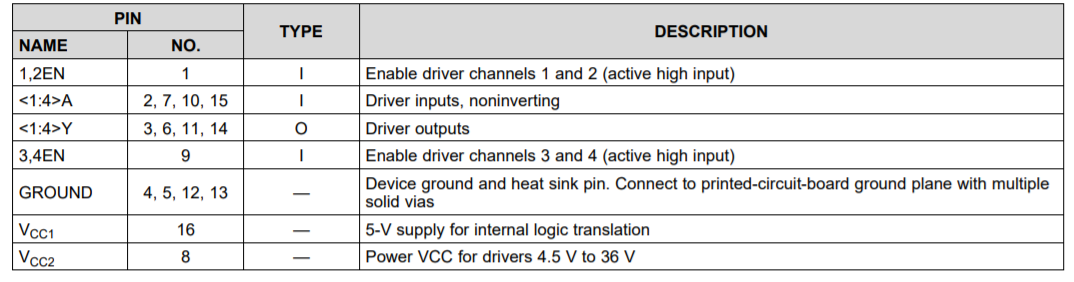
A DC motor converts electrical energy into mechanical energy. You can use a DC motor with an integrated rotary encoder in order to measure the speed of the motor while it’s running. The pinout of the motor are shown in figure 2. Power is applied between pins 1 and 6 of the motor. This power can be drawn directly from a DC power supply or through a motor drive connected to a microcontroller in order to control the speed and the direction of rotation of the motor. The encoder power pin (pin 3) and ground (pin 4) should be connected to a voltage source. The output of the encoder are pins 2 and 5 (Channel A and B respectively). [2]

* L293D Motor Drive:

**FIGURE 3:** L293D IC



L293D is a dual H bridge motor drive. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5V to 36V. The L293D has two +V pins (pins 8 and 16). The pin +VCC2 (pin 8) provides the power for the motors, and +VCC1 (pin 16) for the chip's logic. As shown in figure 4 of the IC pinout. [3] The functions of each pin can be shown in table 1.



**TABLE 1:** L293D IC PINOUT FUNCTIONS

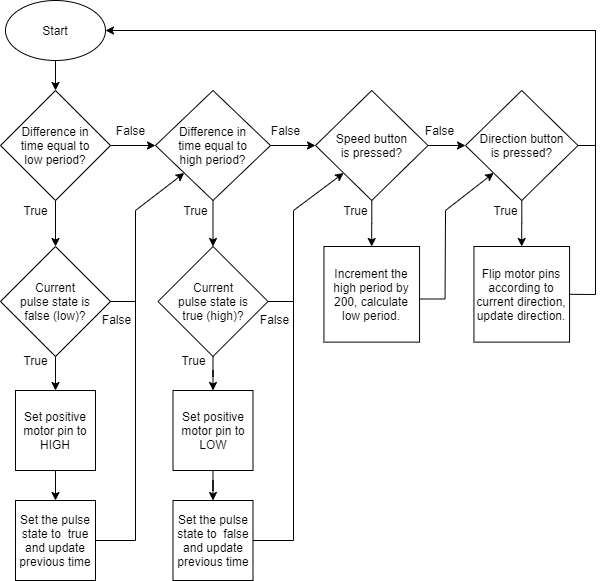
* + 1. **Code Components**
* const: A variable qualifier that modifies the behavior of the variable, making a variable "read-only" and constant.
* long int: Long variables are extended size variables for number storage, and store 32 bits (4 bytes) instead of the usual 16 bits (2 bytes).
* unsigned int: Same as int in that it stores a 2-byte value. Instead of storing negative numbers however they only store positive values, yielding a useful range of 0 to 65,535.
* void setup: The function is called whenever the program starts. It is used to initialize variables, pin modes, etc. It will only run once after each powerup or reset of the Arduino board.
* pinMode(): A function used to configure a specific pin to behave either as an input or an output, usually in the void setup.
* void loop: The function that holds the code inside and runs over and over as long as the board is turned on.
* micros() Returns the number of microseconds since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 70 minutes.  
    
  1. **Methods**
     1. **Code Description**

Once the Arduino Uno turns on, it sets up pins 2 and 4 as output for the positive and negative pins of the motor respectively, and pins 7 and 8 as pullup inputs for the pushbuttons being used to control the speed and direction without the need of resistors. Also, the total period of a pulse is set to 2000 microseconds, which equates to a frequency of 500Hz. The duty cycle is set to 0 microseconds since the motor won’t be rotating initially. A Boolean variable is created to check the current state of each cycle in the PWM signal, and updates whether the pulse is high or low, it’s initially set to false to indicate that it was low. Another Boolean is created to check the direction of the motor, whether clockwise or counterclockwise, it’s initially set to clockwise and updates which each button press.

In the void loop, the current time is updated in every loop in microseconds, since the time intervals are very small for each cycle. The difference between the current time and the last time the pulse signal changed from low to high or vice versa is calculated in two if statements. The first if statement is responsible for the high signal or duty cycle, it checks if the time that passed since the last signal change is equal to the time taken for the low signal period, then checks if the previous state was low and switches it to high by setting the positive pin of the motor to HIGH, then updates the previous time and the current state of the signal to true. The second if statement is responsible for low signal, it follows the same concept but turns off both motor pins setting them to LOW if the conditions are met.

For speed control, an if statement checks if the first pushbutton is pressed, then increments the time taken for the duty cycle by 200 microseconds, which allows 10 levels of speed until the maximum of 2000 microseconds is reached. For directional control, an if statement checks if the second button was pressed and flips the motor pins according to the current direction, then updates the direction. Finally, a debounce function is created to read from each pushbutton instead of using digitalRead to avoid pressing the buttons multiple times, with a debounce interval of 100 milliseconds.

A flowchart is shown describing the code process in Figure 4.



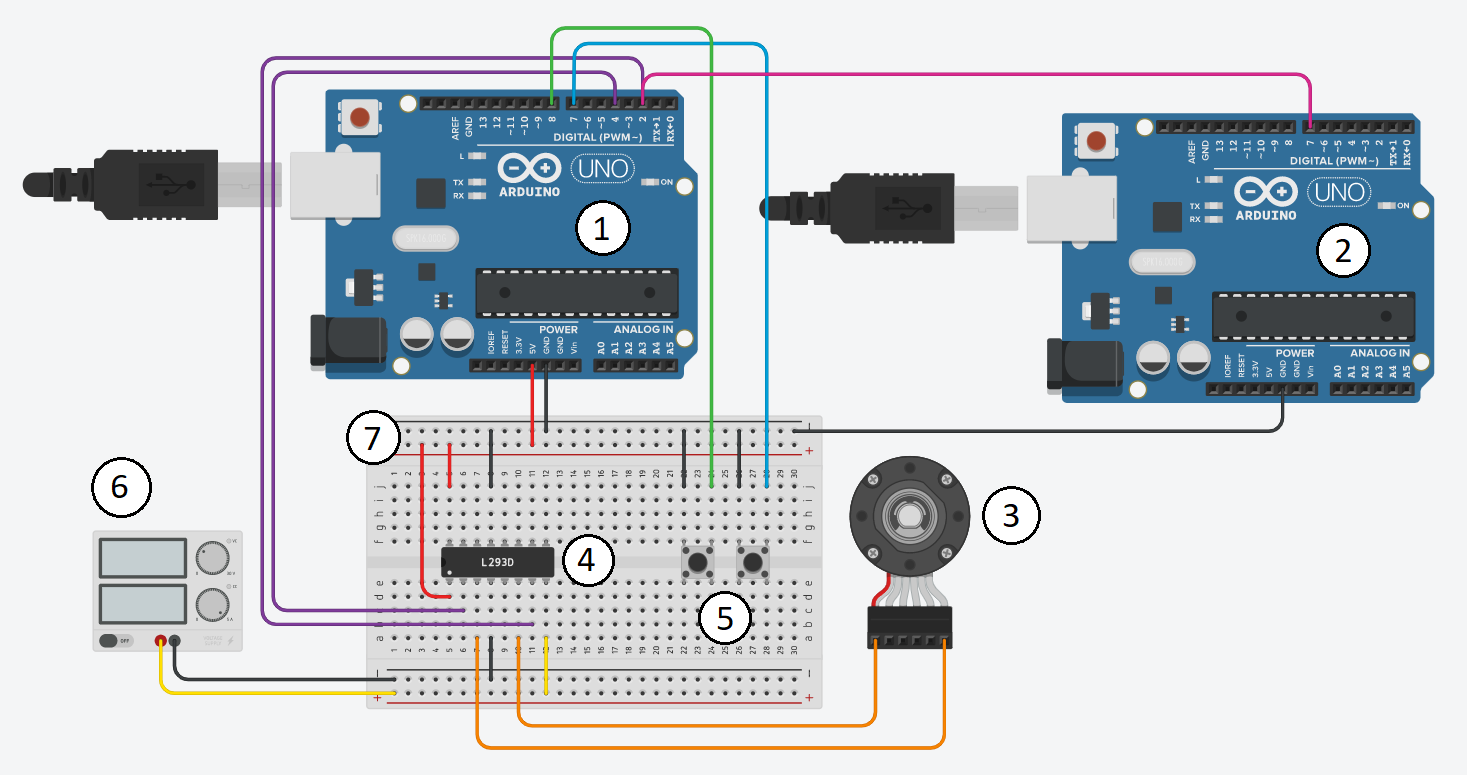
**FIGURE 4:** FLOWCHART

* + 1. **Connections on Tinker CAD**

1. First, connect the 5V and ground pins of the Arduino Uno to the breadboard power and ground (red and black wires), and the power 1, enable, and ground of the L293D to the 5V and ground of the Arduino through the breadboard (red and black wires).
2. Connect the positive and negative pins of the motor to output 1 and output 2 of the L293D respectively (orange wires) to control the direction of rotation of the motor.
3. Connect pins 4 and 2 of the Arduino Uno to input 1 and input 2 respectively (purple wires), and the power supply positive and negative to power 2 and ground of the L293D respectively (yellow and black wires).
4. Finally, connect the first push button to pin 7 and ground of the Arduino (blue and black wires), and the second pushbutton to pin 8 of the Arduino (green and black wires).
5. Due to TinkerCAD’s limitations, a second Arduino is needed to preview the serial plotter and read the cycles of the PWM on a plot, so connect pin 7 of the slave Arduino to pin 2 of the master Arduino to read the outputs to the motor and plot the PWM signal graph.
6. **RESULTS AND DISCUSSION**
   1. **Simulation**

The circuit is implemented on Tinker CAD, adding the code to the Arduino Simulation, obtaining the circuit shown in Figure 5.[4]

**TABLE 1:** LEGEND

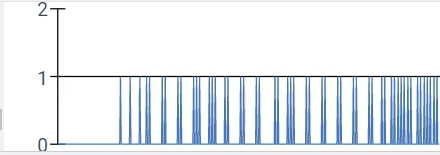


**FIGURE 5:** CIRCUIT SCHEMATIC ON TINKER CAD

|  |  |
| --- | --- |
| NUMBER | COMPONENT |
| 1 | Master Arduino Uno Board |
| 2 | Slave Arduino Uno Board |
| 3 | DC Motor |
| 4 | L293D Motor Drive |
| 5 | Pushbuttons |
| 6 | Power Supply |
| 7 | Breadboard |

* 1. **Analysis**

After running the simulation, the motor is off since the duty cycle is initially set to zero and the serial plotter graph of the slave Arduino is seen as a constant line at 0. After pressing the pushbutton responsible for controlling the speed, 200 microseconds are incremented to the duty cycle time. The motor starts rotating at a low RPM, and the serial plotter graph starts to pulsate from 0 to 1. The more the pushbutton is pressed, the faster the motor rotates and the longer the duty cycle is seen on the graph. After 10 presses, the motor reaches the highest RPM, and the graph is constant at 1 or high. When the second pushbutton is pressed, the motor starts rotating in the opposite direction while keeping the same speed. The graph of the serial plotter measuring the duty cycle can be seen in both figures 6 and 7, initially zero then faster with each button press until it reaches a stable high state of 1 after 10 button presses. The video of the running simulation can be seen through [this link](https://drive.google.com/file/d/1yto1GYSaREjZvNB0LUCH8Elo1udRu9uk/view?usp=sharing).



**FIGURE 6:** INITIAL PWM SIGNAL (2 PRESSES)

**FIGURE 7:** FINAL PWM SIGNAL (10 PRESSES)

1. **CONCLUSION**

The objective of this experiment is to generate a PWM signal out of a digital pin with a frequency of 500Hz to control the speed of a motor without the use of analog input. Two pushbuttons were used to increase the speed and control the direction of rotation of the motor. The experiment resulted in a graph that varies the duty cycle with respect to time according to each button press, and the motor’s speed was controlled using a digital pin with a soft PWM signal successfully.

**REFERENCES**

[1] Pololu Robotics & Electronics. *Arduino Uno*, 2020

https://www.pololu.com/product/2191#:~:text=Overview,header%2C%20and%20a%20reset%20button.

[2] HowToMechatronics, *How Rotary Encoder Works and How To Use It with Arduino*

https://howtomechatronics.com/tutorials/arduino/rotary-encoder-works-use-arduino/

[3] Adafruit, *L293D*

https://learn.adafruit.com/adafruit-arduino-lesson-15-dc-motor-reversing/lm293d

[4] Autodesk Tinkercad  
 https://www.tinkercad.com/

**Appendix**

* + - * Master Arduino code:

const int increment = 200;

int periodT = 2000, periodH = 0, periodL = 2000;

int motorP = 2, motorN = 4;

bool stateHigh = false;

bool clockwise = true;

unsigned long previousmicros = 0;

unsigned long previousDebounceTime = 0;

unsigned long debounceDelay = 100;

void setup()

{

pinMode(8, INPUT\_PULLUP);

pinMode(7, INPUT\_PULLUP);

pinMode(motorP, OUTPUT);

pinMode(motorN, OUTPUT);

digitalWrite(motorP, LOW);

digitalWrite(motorN,LOW);

}

void loop()

{

unsigned long currentmicros = micros();

if(currentmicros - previousmicros >= periodL){

if(stateHigh == false){

digitalWrite(motorP, HIGH);

digitalWrite(motorN, LOW);

stateHigh = true;

previousmicros = currentmicros;

}

}

if(currentmicros - previousmicros >= periodH){

if(stateHigh == true){

digitalWrite(motorP, LOW);

digitalWrite(motorN, LOW);

stateHigh = false;

previousmicros = currentmicros;

}

}

if(debounce(7) == LOW && periodH < periodT){

periodH += increment;

periodL = periodT - periodH;

}

if(debounce(8) == LOW){

if(clockwise == true){

motorP = 4;

motorN = 2;

clockwise = false;

}

else{

motorP = 2;

motorN = 4;

clockwise = true;

}

}

}

int debounce(int buttonPin){

int reading = digitalRead(buttonPin);

if (reading == LOW) {

if ((millis() - previousDebounceTime) > debounceDelay) {

previousDebounceTime = millis();

return LOW;

}

}

return HIGH;

}

* + - * Slave Arduino code:

int readPin = 7;

void setup() {

Serial.begin(19200);

pinMode(readPin, INPUT);

}

void loop(){

Serial.println(digitalRead(readPin));

}