Task-1: BLDC Commutation Using 3 Half H-Bridges

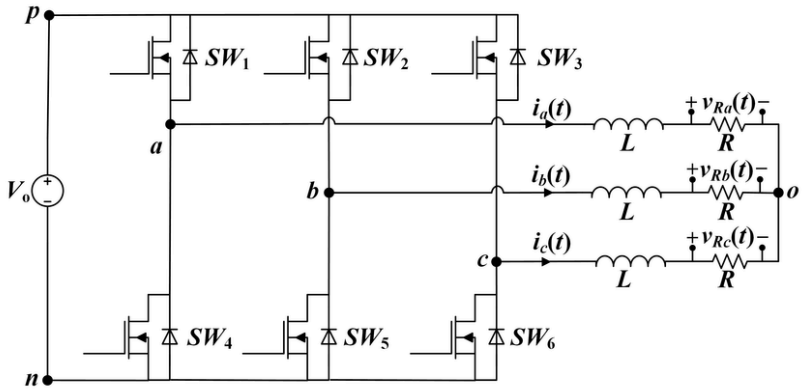
Rishith Susarla
23EC10068

Objective: Coding a Brushless DC motor for six-step commutation, using an Arduino Uno, a 3-leg half H-bridge, and a BLDC motor

Modules:

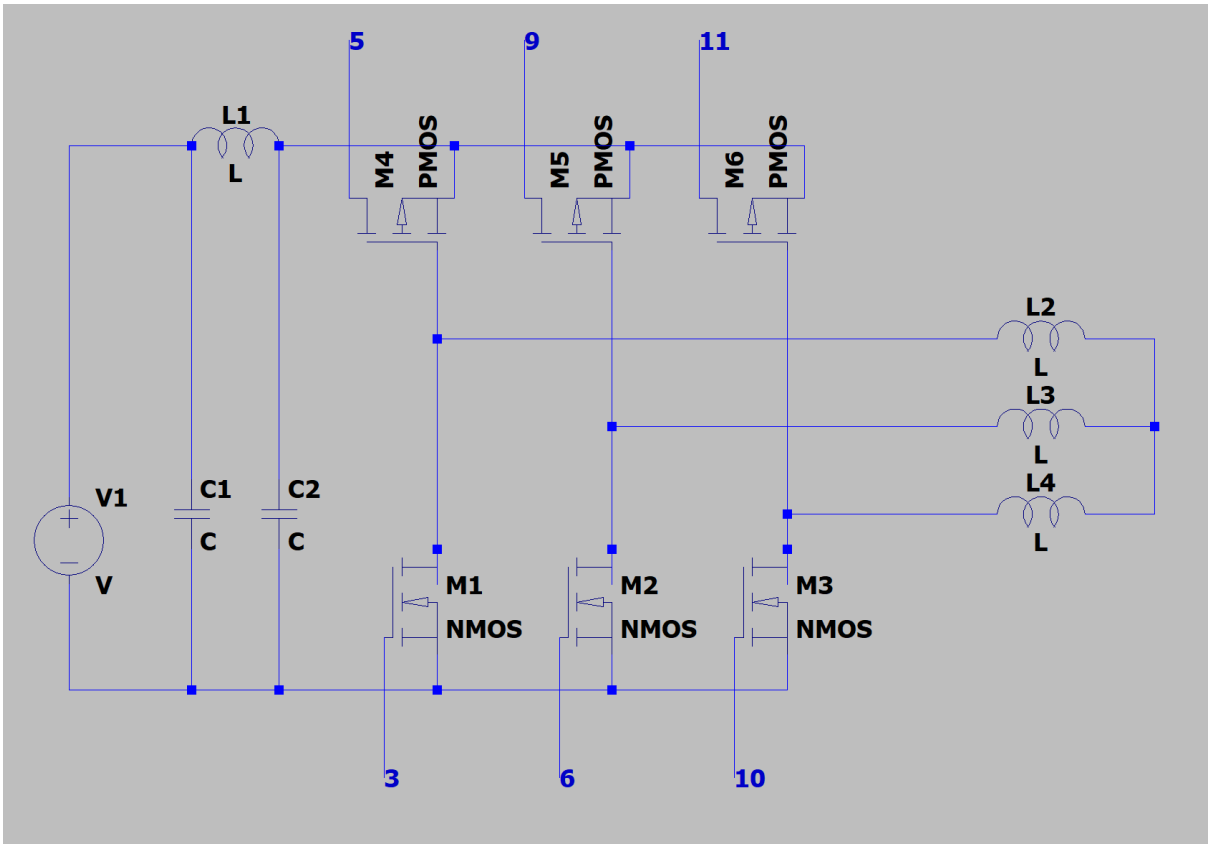
1) **Arduino UNO:** The Arduino UNO here is used as the controller of the circuit. It provides the inputs for the MOSFETs to switch on and off to control the phases of the inductor coils present in the Brushless DC Motor.

2) **3-Leg Half H-Bridge:** Each Half H-Bridge is used to control an inductor coil. We use MOSFETs as switches to control the current passing through these coils. For the below schematic, we use PMOS for high side switching and NMOS for low side switching. For a NMOS, we need a HIGH signal at the gate to turn on while for a PMOS, a LOW signal turns on the gate.



3) **Brushless DC Motor:** A brushless DC motor operates on a DC voltage supply and uses electric communication to rotate rather than traditional brushes found in conventional DC motors.
Brushless DC motors consist of inductor coils (stators) and one permanent magnet placed in between (rotor). They utilise the concept of magnetic field produced by a current carrying conductor to cause the rotor to rotate accordingly. The more inductor coils used, the more the accuracy of rotation will be. Hall Sensors may be used to determine the position of the rotor. Hence, according to the position of the rotor, different inductor coils are excited to cause the rotor to rotate.

Schematic:



Truth Table:

Task-1 Truth Table

The link below contains the code and schematics I made for the task.

Task-1

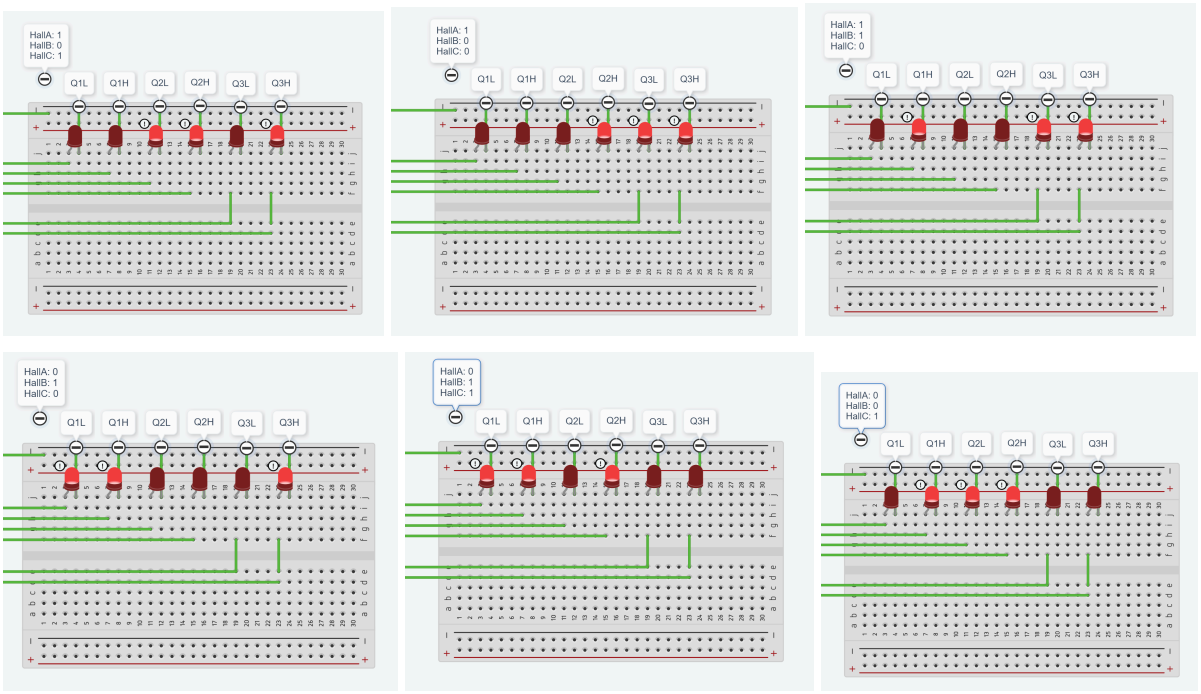
Code:

```
// Hall Sensors
#define H1_pin 4
#define H2_pin 7
#define H3_pin 8

// MOSFETs
#define Q1L 3
#define Q1H 5
#define Q2L 6
#define Q2H 9
#define Q3L 10
#define Q3H 11

int i, topSpeed = 10, speed = 5;

int PWMOutput = int(map(speed, 0, topSpeed, 0, 255)); // PWM Output to control speed of
the motor

volatile bool HallA, HallB, HallC;

void setup() {
  pinMode(H1_pin, INPUT);
  pinMode(H2_pin, INPUT);
  pinMode(H3_pin, INPUT);

  pinMode(Q1L, OUTPUT);
  pinMode(Q1H, OUTPUT);
  pinMode(Q2L, OUTPUT);
  pinMode(Q2H, OUTPUT);
  pinMode(Q3L, OUTPUT);
  pinMode(Q3H, OUTPUT);

  HallA = digitalRead(H1_pin);
  HallB = digitalRead(H2_pin);
  HallC = digitalRead(H3_pin);

  PCICR != B00000101; // Activating interrupts for groups 0 and 2
  PCMSK2 != B10010000; // pins 4,7 will trigger interrupts
  PCMSK0 != B00000001; // pin 8 will trigger interrupt
  //Serial.begin(9600);

}

ISR (PCINT2_vect){ // Interrupt Service Routine (ISR) for pins 4,7
  HallA = digitalRead(H1_pin);
  HallB = digitalRead(H2_pin);
  HallC = digitalRead(H3_pin);
}

ISR (PCINT0_vect){ // Interrupt Service Routing (ISR) for pin 8
  HallA = digitalRead(H1_pin);
  HallB = digitalRead(H2_pin);
```

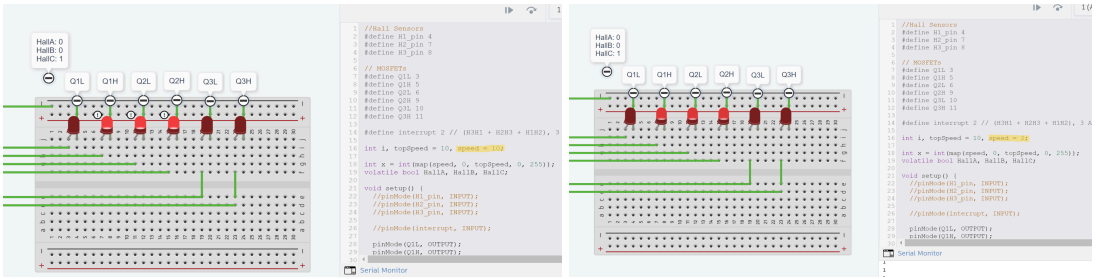
```
  HallC = digitalRead(H3_pin);
}

void loop() {
  // Providing the pins with voltage according to the Hall Sensor Readings
  analogWrite(Q1L, PWMOutput*getQ1L(HallA, HallB, HallC));
  analogWrite(Q1H, PWMOutput*getQ1H(HallA, HallB, HallC));
  analogWrite(Q2L, PWMOutput*getQ2L(HallA, HallB, HallC));
  analogWrite(Q2H, PWMOutput*getQ2H(HallA, HallB, HallC));
  analogWrite(Q3L, PWMOutput*getQ3L(HallA, HallB, HallC));
  analogWrite(Q3H, PWMOutput*getQ3H(HallA, HallB, HallC));

  delay(200);
}


// Functions to get the switch value by using a logic expression derived from the Hall sensor inputs
bool getQ1L(bool H1, bool H2, bool H3){
  return (H2 && !H1);
}

bool getQ1H(bool H1, bool H2, bool H3){
  return ((!H3 && H2) || (H3 && !H1));
}

bool getQ2L(bool H1, bool H2, bool H3){
  return (H3 && !H2);
}

bool getQ2H(bool H1, bool H2, bool H3){
  return ((!H2 && H1) || (H3 && !H1));
}

bool getQ3L(bool H1, bool H2, bool H3){
  return (H1 && !H3);
}

bool getQ3H(bool H1, bool H2, bool H3){
  return ((!H3 && H2) || (H1 && !H2));
}
```

Simulation Results:

- **Testing Switch Logic Functions:** To test the logic of the switches, I used TinkerCAD and made the connections accordingly such that the bulb glows when the logic corresponds to HIGH. By changing the Hall Sensor inputs, I compared the results with the truth table to ensure the logic was right.



- **Testing PWM Logic for Speed of Motor:** To test the PWM logic for the speed of the motor, I used the same circuit and checked to see whether the brightness of the bulb altered, which it did clearly as shown.



Logic:

Aim: To code six step commutation of a BLDC motor

The BLDC motor (3-phase) consists of 3 inductor coils (stator) and 1 permanent magnet (rotor). According to the position of the rotor, we excite different stator coils to cause the rotor to rotate using principles of attraction and repulsion of magnets.
Hence, we first take in the inputs of the Hall Sensors to get an idea of the position of the rotor. Given the Hall Sensor readings at a specific moment, we can figure out the states of all the inductor coils. Using the 3-leg Half H-bridge, we accordingly set the states of the switches to achieve the required orientation.
To control the speed of the rotation of the BLDC motor, we need to increase the current provided to the three stator coils. The more current passed, the stronger the magnetic field will be to cause the rotation of the rotor. Hence, the MOSFETs are connected to the PWM pins to be able to regulate the voltage provided to the coils.

In the Arduino code setup, first we set the Hall Sensor pins to be the inputs and the PWM pins connected to the MOSFETs to be the outputs. Then, we take the initial inputs of the hall sensors from the respective pins.

I used pin change interrupts such that any change of value at the Hall Sensor pins will trigger the interrupt and the readings of the hall sensors will be taken again.

According to the truth table devised for the states of the switches, we derived a logic expression and wrote functions accordingly. Initially, according to the speed desired by the rotor, we derived a PWM Output to correspond to the high state by using a mapping function to get a value in the range of 0-255.
In the loop of the code, to the pins connected to the MOSFETs, we gave the output to be the PWM Output multiplied by the result of the logic expression for the corresponding switch according to the hall sensor values.

Difficulties Faced:
I wasn't able to find a proper way of simulating the code anywhere. I tested the logics I derived from Karnaugh's maps using LEDs as shown above and checked the PWM part by controlling the brightness of an LED but as I lacked

a proper simulating setup, I wasn’t able to test the smooth rotation of the motor and also the interrupts.