

HNC in Electrical and electronic engineering

Micros Peripherals and Interfaces.

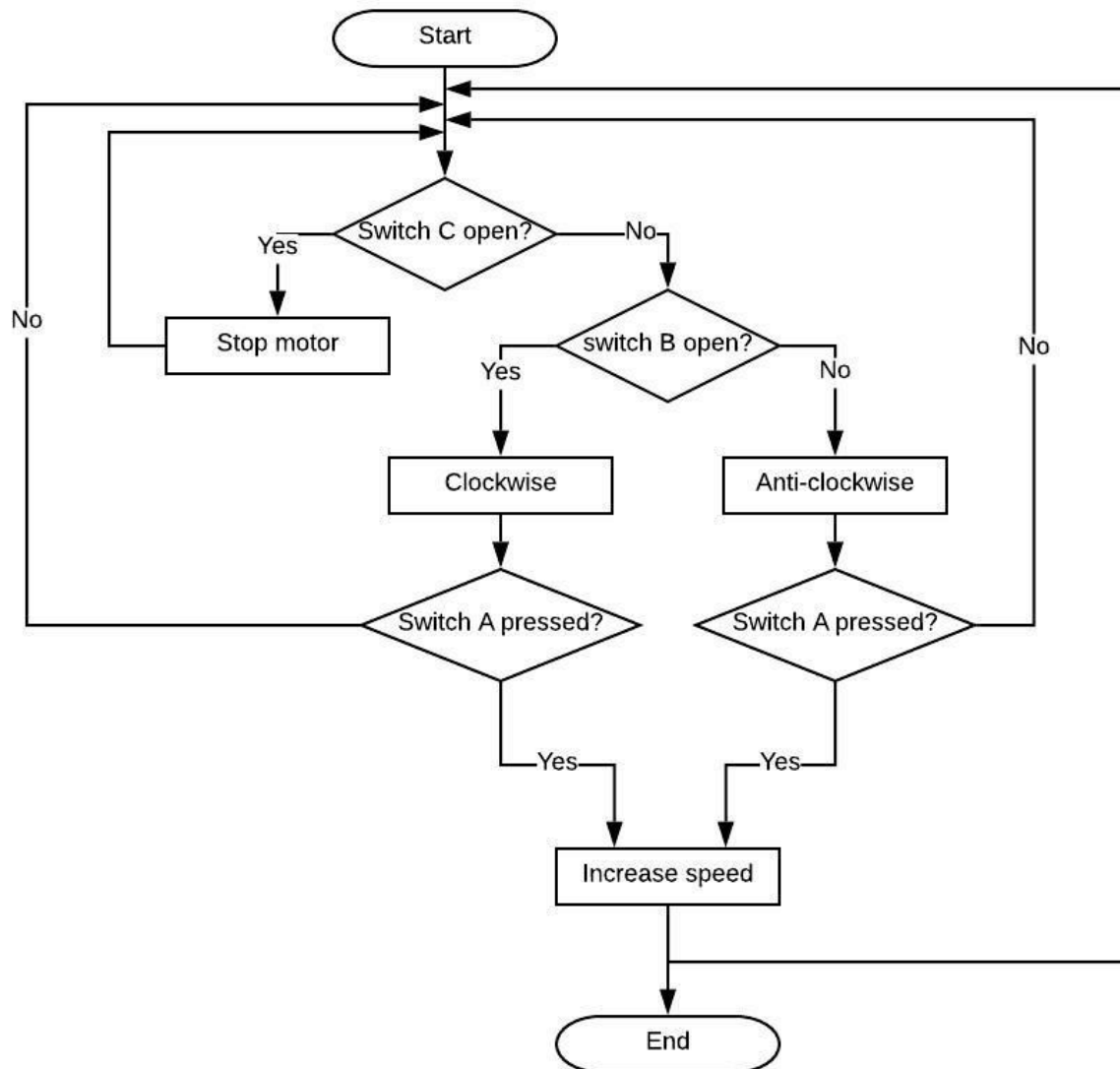
Micro controllers.

By Aneurin Creed

2018/19

Task 1 Motor Control

Using lucidchart i have created a flowchart that depicts sequence of events that the program will run through.



I've used Arduino to write the code for this program. Arduino's programming language is made up of three main groups, Values, Structure and Functions. I've used a combination of functions from these groups to produce a programme that will control the motor as required.

Use of the if function has allowed me to set a stimulus criteria which when met produces a predefined automated response. For example, when switch C is in the closed position, it allows the motor to run. However, at any point during the run of the programme, the motor can instantly be stopped by the switch being triggered.

```
const int pwPin      =6;
const int enPin      =7;
int A                =2;
int B                =3;
int C                =4;
int speedo(1500);

void setup() {
  pinMode  (A,    INPUT);
  pinMode  (B,    INPUT);
  pinMode  (C,    INPUT);

  pinMode (pwPin, OUTPUT);
  pinMode (enPin, OUTPUT);
  Serial.begin(9600);
  while (!Serial)
    Serial.println("press A to increase speed");
  Serial.println();
}

void loop() {
  if(digitalRead(C)==HIGH){
    if(digitalRead(B)==HIGH){
      if(digitalRead(A)==HIGH){
        Serial.println("A");
        delay(1000);           // stops overcounting
        Serial.println("increase speed");
        speedo=speedo+50;
        Serial.println("speed=");
        Serial.println(speedo-1500);
      }
    }
    if(digitalRead(B)==LOW){
      if(digitalRead(A)==HIGH){
```

```

        Serial.println("A");
        delay(1000);           // stops overcounting
        Serial.println("increase speed");
        speedo=speedo-50;
        Serial.println("speed=");
        Serial.println(speedo-1500);
    }
}

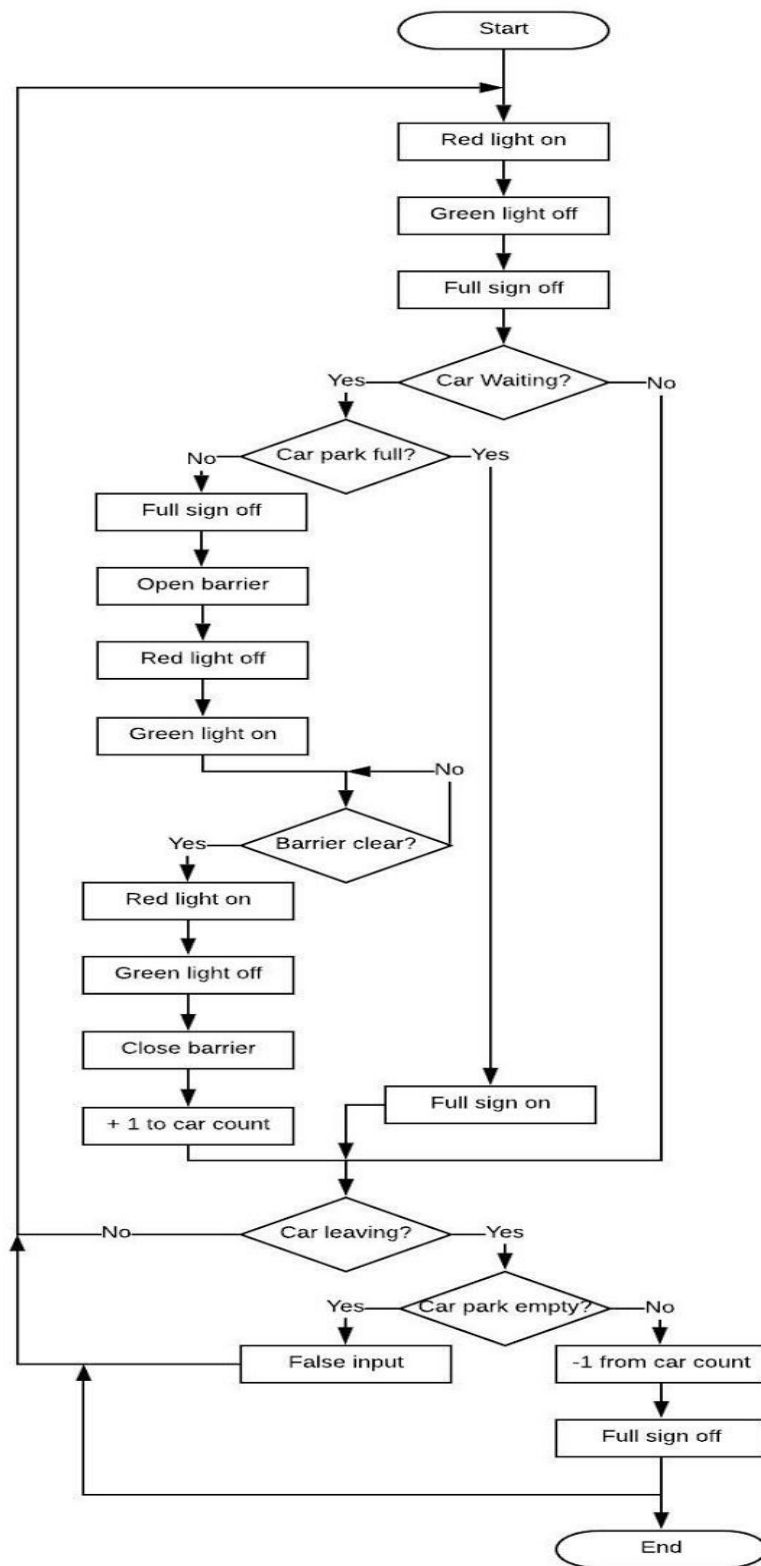
if(digitalRead(C)==LOW){
    speedo=1500;
}
    digitalWrite(pwPin,HIGH);
    delayMicroseconds(speedo);
    digitalWrite(pwPin,LOW);
    delay(20);
}

```

Even though the programme achieves the requested criteria, an issue that I am aware of is that once the maximum speed has been reached, the user is still able to press the push switch to instigate an increase in speed. However, the digital output of the speed does increase the physical rpm does not go beyond maximum. This limitation indicates a lack of measured response within the system, thus producing a false reading. To overcome this, I could embed a maximum limit reached message that would be presented to the user indicating that the speed can not be increased further.

Task 2 car park

As per my previous flowchart, I also used lucidchart to design this for my programme.



Using Arduino, I have created a programme within which I have embedded a pre-existing example to run the stepper motor. The example appears within the code as a sub-routine.

```
#include <Stepper.h>
```

```
const int stepsPerRevolution = 50;
```

```
Stepper myStepper(stepsPerRevolution, 8, 11, 12, 13);
```

```
int count =0;
```

```
int full =3;
```

```
int S1    =2;
```

```
int S2    =3;
```

```
int S3    =4;
```

```
int green  =5;
```

```
int red    =6;
```

```
int sign   =7;
```

```
void setup() {
```

```
count = 0;
```

```
full = 2;
```

```
    myStepper.setSpeed(60);
```

```
    Serial.begin(9600);
```

```
pinMode    (S1,  INPUT);
```

```
pinMode    (S2,  INPUT);
```

```
pinMode    (S3,  INPUT);
```

```
pinMode    (green, OUTPUT);
```

```
pinMode    (red,  OUTPUT);
```

```
pinMode    (sign, OUTPUT);
```

```
digitalWrite (green, LOW);
```

```
digitalWrite (red,  HIGH);
```

```
digitalWrite (sign, LOW);
```

```
pinMode    (9,   OUTPUT);
```

```
pinMode    (10,  OUTPUT);
```

```
digitalWrite (9,  HIGH);
```

```
digitalWrite (10, HIGH);
```

```
    }
```

```

void loop() {
    if(digitalRead(S1)==HIGH){
        Serial.println("Car waiting at S1");
        delay(1000); // stops overcounting
        if((count)==full){
            Serial.println("Car cannot enter");
            digitalWrite(sign,HIGH);
            Serial.println("full Sign on");
        }
        if((count)<full){
            digitalWrite(sign,LOW);
            Serial.println("full Sign off");
            Serial.print("Count= ");
            Serial.println(count);
            myStepper.step(50);
            Serial.println("Barrier open");
            digitalWrite(red,LOW);
            Serial.println("Red off");
            digitalWrite(green,HIGH);
            Serial.println("Green on");
        }
    }
    if(digitalRead(S2)==HIGH){
        Serial.println("S2");
        delay(1000); // stops overcounting
        if((count)==full){
            Serial.println("False Input");
            digitalWrite(sign,HIGH);
            Serial.println("full Sign on");
        }
        if((count)<full){
            digitalWrite(red,HIGH);
            Serial.println("Red on");
            digitalWrite(green,LOW);
            Serial.println("Green off");
            myStepper.step(-50);
            Serial.println("Barrier close");
            count++; // add 1 to count
            Serial.print("Count= ");
            Serial.println(count);
            delay(1000);
        }
    }
}

```

```

if(digitalRead(S3)==HIGH){
    Serial.println("S3");
    delay(1000);                // stops overcounting
    if((count)==0){
        Serial.println("False Input");
    }
    if((count)>0){
        count--;                // minus 1 from count
        digitalWrite(sign,LOW);
        Serial.println("Sign off");
        Serial.print("Count= ");
        Serial.println(count);
        delay(1000);
    }
}
}

```

A major shortcoming of this programme is the distinct difference between the model and the real life scenarios that it will face such as vehicle diversity. In my setup each switch would register one push as an input whereas generally cars have two sets of wheels causing two inputs per switch, also different vehicles have different number of wheels and will each register differently. To overcome this a different method of detection would have to be in place and then be configured. Something like light gates would work perfectly here as this would only register the physical presence of a vehicle or lack of as the signal for the system.

Another fault within the programme is that sensor 2 can be triggered without sensor 1 being previously engaged. This would result in incorrect readings for the counter and possibly cause the programme to crash. Furthermore, a worse outcome that could occur would be that there is a mechanical failure where motor drives past its rest point causing damages. To solve this, code would need to be written that disables any reading of sensor 2 unless sensor 1 has been triggered first.