Aylmer Britto R

Technocrat's summer vacation task report-II(Week 2)

ENCODERS:

In simple words encoders are position sensors. An encoder is a sensor of mechanical motion that generates digital signal in response to mechanical motion like motion of motors. Encoders can be classified into absolute encoder (for tracking the covered path) and incremental encoder (to track velocity and direction and also it is referred as quadrature encoder). In further encoders can also be classified into rotary encoder (for rotational motion) and linear coder (for linear motion).

Incremental encoders:

In case of incremental encoders a single channel output is used to study only the speed of the motion. Here there is transparent disc with equally spaced lines with opaque substances. There is light source on one side and there is a photo-detector on the other side. When the disc is made to rotate by external force there is a square pulse generated based on the light passing through the disc. From the pulse generated the pulse width is observed and the speed can be manipulated accordingly. When the direction also has to be studied two channel output comes into play. The output pulse of the photo-sensors are obtained. Based on the phase shift produced between the two signals the direction of rotation is obtained.

Encoders are classified on two basis. Based on sensing technology and output. Based on sensing technology it is classified as magnetic, optical and laser. Based on output produced they are classified as incremental encoders and absolute encoders

Absolute encoders:

Say in case of stepper motors there might be a necessity where the position motor has to be obtained. In that case we go for absolute encoders. Absolute encoders are required when the position of the motor also matters. The spaces in the disk follow a definite pattern. A counter is there to count the number of lines passed. They are designed in such a way that it has 2^n markings for every n th circle. This implies that precision is upto 1/n times per rotation.

(ADD IMAGE)

It is best to choose magnetic encoder over linear encoder because of the disturbance caused by the interference of dust particles, oil and light has no effect on the environment.

The principle behind operation of magnetic encoders are hall-effect and magneto-resistive. The magnetic encoders have a rotor and a sensor. The rotor has alternate magnetic poles in a disc like a pie-chart.

Interfacing the encoder module and sensor:

There are five pins in the rotary encoder module. Vcc, Gnd, outputA, outputB and a button pin. Hopefully the first four pins are known to us.

Button Pin:

In the encoder there are switches that goes on and off frequently. When they oscillate between this on and off they switch may bounce causing minor disturbance in the signal. So there arises a situation these bounces have to be debounced. There are two types of debouncing(Hardware and software). In hardware debounce a combination of resistors and capacitors are used to debounce the internal noise. When arduino and some other programmable micro-controllers are used the debouncing can be done in the programming part itself(software debouncing) eliminating the need for hardware components. In fact there are pre-defined library for debouncing in arduino IDE.

The following is the sample code for

```
#define outputA 6
#define outputB 7
int counter = 0;
int aState;
int aLastState;
void setup() {
  pinMode (outputA,INPUT);
  pinMode (outputB,INPUT);
```

```
  Serial.begin (9600);
  // Reads the initial state of the outputA
  aLastState = digitalRead(outputA);
}
void loop() {
  aState = digitalRead(outputA); // Reads the "current" state of the outputA
  // If the previous and the current state of the outputA are different, that means a Pulse has occured
  if (aState != aLastState){
    // If the outputB state is different to the outputA state, that means the encoder is rotating clockwise
    if (digitalRead(outputB) != aState) {
      counter ++;
    } else {
      counter --;
    }
    Serial.print("Position: ");
    Serial.println(counter);
  }
  aLastState = aState; // Updates the previous state of the outputA with the current state
}
```

PNEUMATIC SOLENOID:

Inner arrangements of the solenoid and its working:

The solenoid consists of coil of wire, armature and a push pin. There is a spring at each end of the solenoid to assist the shaft in the middle to retract to the centre position when in neutral condition. When the power supply is given to the solenoid (High signal from the micro-controller) a electromagnetic field is created. The field exerts a force on the armature such that it pulls the armature in the direction of the winding. This makes a way for the air to flow through the port.

Setting up the system:

One part of the solenoid has three ports(say side A) and the other side has two ports(say side B). Through the middle port in side A the air enter from the air compressor and the other two ports are the exhaust. Exhaust mufflers are used in the exhaust ports to minimize the sound. The exhaust mufflers can be adjusted in such a way that it controls the speed of vibration. Two pipes connected from the side B ports are connected to the ports of the cylinder(the actual actuator where the shaft is attached) based on the necessity. Before connecting the port in the cylinder

the decision should be made about the default position of the cylinder(whether it should be in extended position or contracted position when it is not powered). Based on the decision the pipe connections are given to the cylinder. There is also mechanical switch to test the working when needed.

Controlling the speed of motion:

By controlling the air flow in the cylinder the speed of the motion can be controlled. There is no necessity for controlling it electrically as the valves can be adjusted easily. The extending speed and the retracting speed can be controlled independently i.e. the system can be designed in such a way that it extends faster and retract back slowly.

Circuit Connections:

The 5V from the Arduino is not sufficient to drive the solenoid so a simple NPN transistor can be used as a switch. The output pulse is given to the base of the base of the transistor (gate in case of MOSFET). The emitter (Drain in MOSFET)is connected to the common ground of the system. The collector terminal is connected to one terminal of the solenoid. And the other terminal of the solenoid is connected to the power source.

Relays can also be used for on/off control.

Coding:

The normal coding which is used for blink program can be used.Giving a 'HIGH' pulse on the gate of the transistor will operate the solenoid. No other complex programming is required.

ACCELEROMETER:

It senses the direction in which the system is oriented. The basic principle behind the operation is such that it senses the acceleration due to gravity from the position it is held and the direction/ angle of inclination is found out. The acceleration due to gravity is something associated with mechanical property to relate it with electrical output MEMS (Micro electrical mechanical Deiveces)technology comes into play. This sensor is actually a capacitive based sensor. The electrically charged plates are separated by a distance and the system contains many arrangements like this. The whole system has a suspended mass at the end. When the system is left to hang the mass experiences a gravitational pull. Due to this pull the electrically charged plates are also pulled, so there is a change in distance between these plates leading to the change in capacitance of the plates. The system is calibrated when kept in exactly horizontal position and compared with other orientations. There is a relation between acceleration due to gravity and the angle of inclination of the system. So the angle of inclination can be found out using the relation.

ADXL345:

The module has 8 pins.

Gnd, Vcc (power supply), CS(chip select), INT1, INT2(interrupt output pin 1 and 2), SDO(serial data output),SDI(serial data input), SCLK(serial communication clock).

The module can communicate in two different protocols (SPI and I2C).

The resolution of the instrument is less than one degree.

SPI Mode:

This mode is possible with 3 and 4 wires. When operated in 3 wire the SDO is connected to power supply or gnd with the pull down resistor. Four 4 wire communication:

GND GND

3V3 VCC

10 CS

12 SDO

11 SDA

13 SCL

In the coding part to select SPI connection:

```
ADXL345 adxl = ADXL345(10);
ADXL345(CS_PIN);
```

I2C mode:

GND GND

3V3 VCC

3V3 CS

GND SDO

A4 SDA

A5 SCL

In the coding part to select I2C connection:

```
ADXL345 adxl = ADXL345();
```

In the ADX library there are inbuilt functions for x y and z axes. They can be called when required.

```
adxl.readAccel(&x, &y, &z)
```

Interrupts:

An interrupt is a high priority signal given by the device to the micro-controller. This signal interrupts the current code the controller is already executing.

ADXL_ISR() This command is for interrupts.

The ADXL345 module offers extra feature like single tap or double tap detection, Activity or inactivity sensing and free fall detection.

PID Controller:

Before going to the controller first of all we will understand what is PID. PID stands for Proportional Integral and Differential. This system is used for reducing the error and correcting it. Let us consider a scenario that an oscillating pointer should stop in an exact position(mean position). When the pointer is made to point at the desired position it might end up in land away from the actual desired position. When the pointer is connected with the proportional factor of the PID the error is corrected proportionally i.e. it oscillates about the mean position(desired position) and does not rest at the exact position. When the differential factor is also added to the system the rate of changing error is considered and corrected. So when the differential factor is considered the oscillation is damped and it stops at the exact desired position. Now the system is accurate but the damping oscillation is not impressive. So now the integral factor comes into play. The steady state error is overcome by this factor. Hereafter there are no more damping oscillations. So, after the integral factor is considered when the pointer is left from the rest position the pointer stops exactly at the desired point.

So from the whole process it is concluded that the PID system gets feedback from the operating device processes it(the error that was already committed, the error it is committing at present and the error that is yet to be committed) and it comes up with a alternative solution and acts on the device again.

Kp-defines how the PID reacts to the current amount of error

Ki- defines how the PID reacts to error over time

Kd-defines how the PID reacts to the change in error

Such a PID control can be interfaced with an arduino too.

Arduino Setup:

The PID controller has 5 pins. The pins are Vcc, Gnd, encoder, signal and MOQ pin.

```
#include <PID_v1.h>

//Define Variables we'll be connecting to
double Setpoint, Input, Output;

//Specify the links and initial tuning parameters
PID myPID(&Input, &Output, &Setpoint,2,5,1, DIRECT);

void setup()
{
  //initialize the variables we're linked to
  Input = analogRead(0);
  Setpoint = 100;

  //turn the PID on
  myPID.SetMode(AUTOMATIC);
}

void loop()
{
  Input = analogRead(0);
  myPID.Compute();
  analogWrite(3,Output);
}
```

Some of the PID functions are listed and explained:

PID ()- It initializes the PID controller . The parameters passed in the function are input, output, setpoint, direction and the tuning parameters like Ki(integral factor), Kp(Proportional factor) and Kd(differential factor).

Compute ()-This is a Boolean return type function. Returns True when the output is computed. Returns False when the output is not manipulated. It should be called once in every loop. It calculates the output at the desired frequency.

SetMode ()-This function is passed to decide whether the PID should operate manually or automatically. The parameter passed should be either 'AUTOMATIC' or 'MANUAL'. By default it is in off state only. This is a void return type function.

SetOutputLimits ():The output ranges from 0-255. The minimum and maximum output value should be passed as parameters. This is a void type return function.

SetTunings (): The parameter passed are the three PID factors Kp, Ki, Kd respectively. This is a void type return function.

SetSampleTime (): Sets time how often the PID must be evaluated. The time period in milli-seconds is passed as a parameter. The time period ny default is 200ms. This is a void type return function.

DisplayFunctions(): There are display functions like GetKp()

, GetKi(), GetKd(), GetMode(). GetDirection().