ELEC3204 Project – The Spin Chamber

Group 3  
Lisa Mobilia (460452436)  
Qizhang Li (470382152)  
Rijul Gupta (470230990)

# Project Description



Based on the Human Centrifuge used by NASA to simulate high g environments for astronaut training. The force that a person sitting in the chamber would experience a force proportional to the square of the speed that the chamber is rotating in, this allows us to simulate high g environments. Cosmologists have long predicted that when we go inhabit space the space colonies would be made of such chambers as they provide us with an artificial sense of gravity and helps us escape weightlessness.

In this project, we have made use of 2 DC Motors set in a feedback loop to simulate a controlled speed spin chamber, to make the chamber we have used TODO which we have mounted on a spur gear. To simulate different loads, we have added TODO.

# Hardware Specifications

## Controls

1. Main/Power switch – used to switch the whole device on/off
2. Mode switch – a potentiometer used to switch between different modes

The Main Switch is driven at half the bus voltage by using a voltage divider to power its high port. This is done to ensure that the output voltage that is fed to ADC of PICAXE is always much less than 5 and hence it does not get damaged.

## Filter

A low pass filter has also been implemented to reduce the noise that is produced by the feedback motor’s output. We have used a Capacitor of 0.08µF and a resistor of

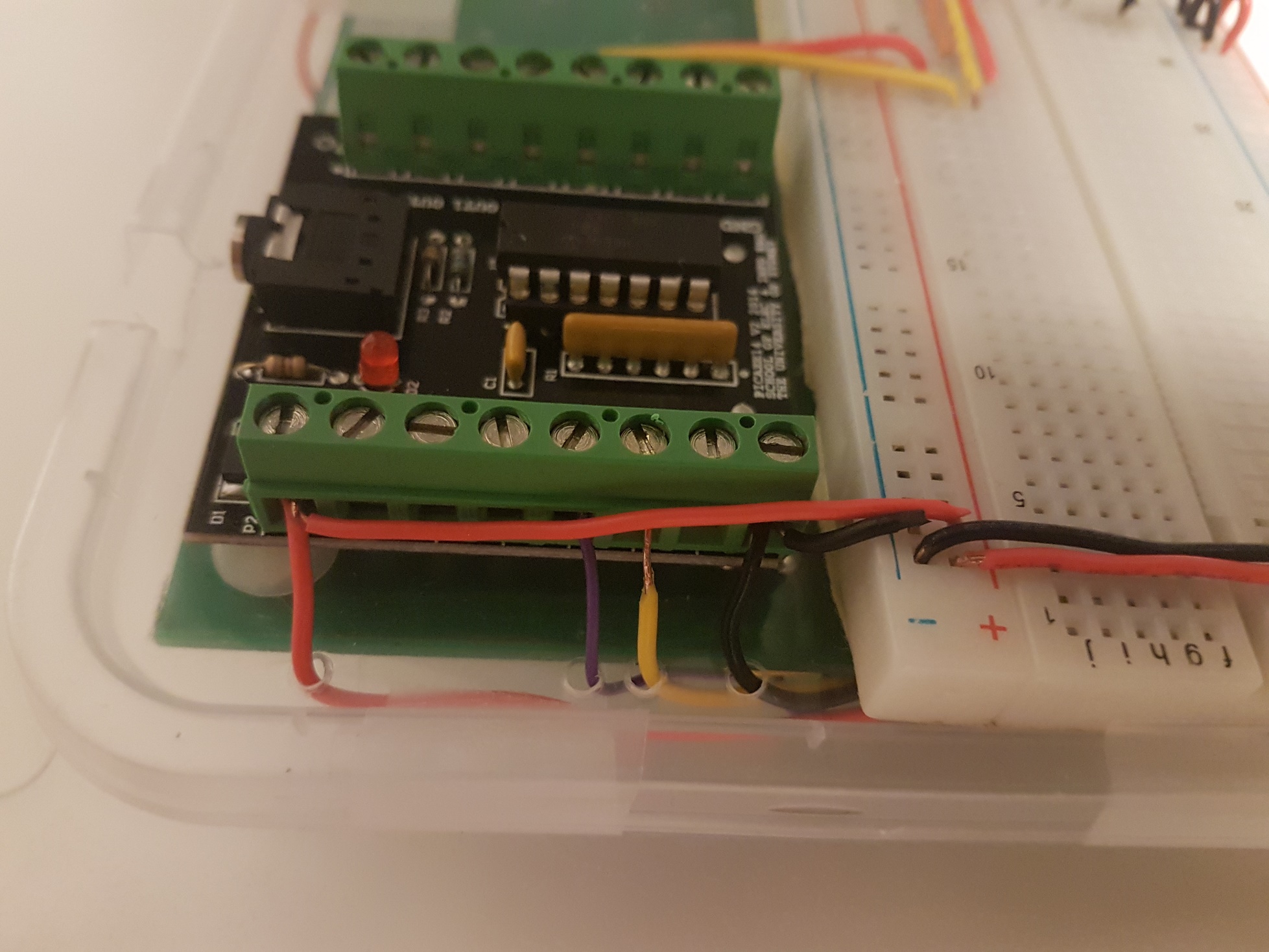
## Connections

### Connections on H BridgeC:\Users\rijul\AppData\Local\Microsoft\Windows\INetCacheContent.Word\20170604_000530.jpg

Color code:

1. Red: 5V
2. Black: Ground
3. Yellow: Q1, Q4
4. Purple: Q2, Q3

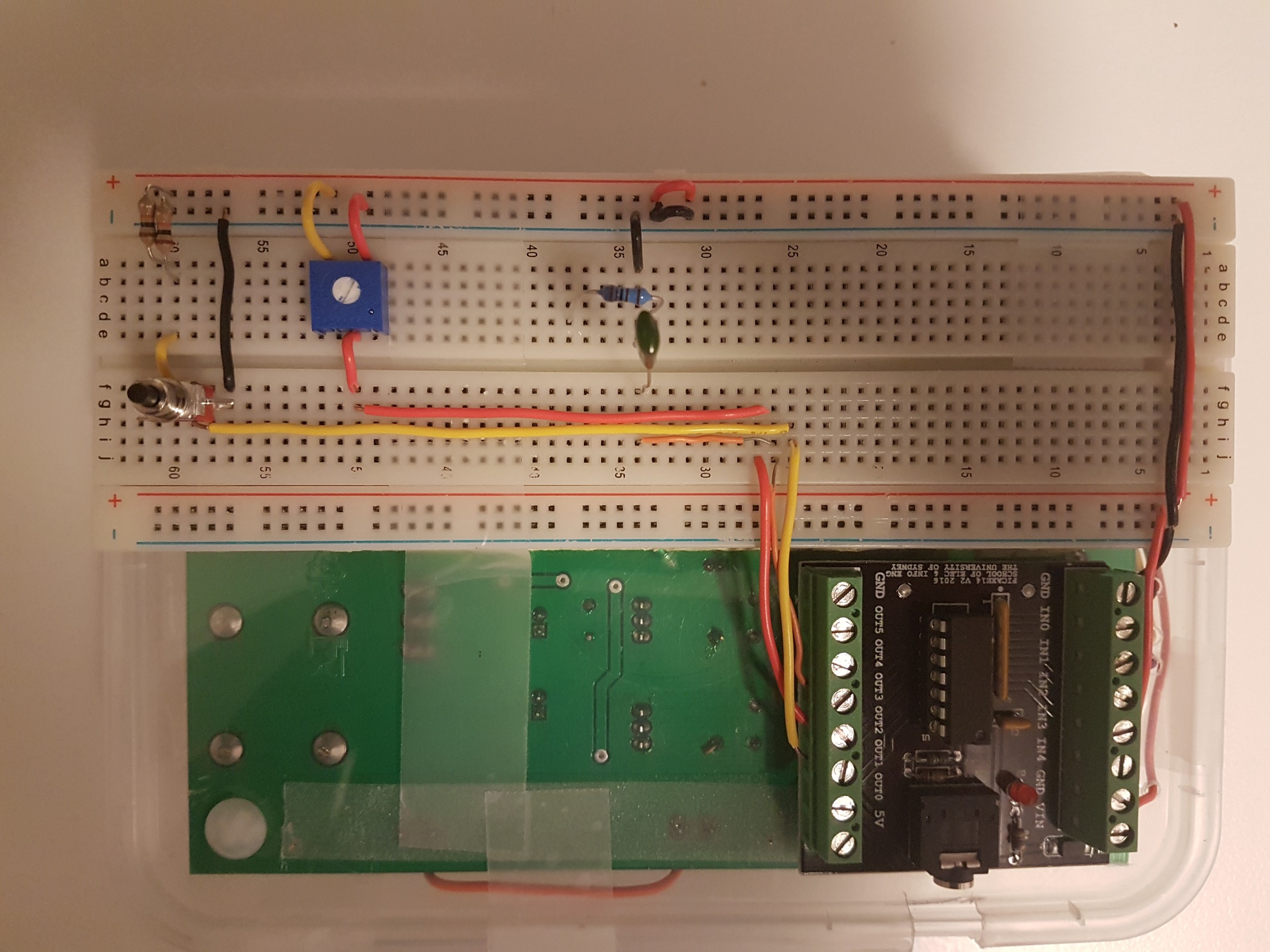
### Connections on PICAXE



Color code:

1. Red: Vin
2. Black: Ground
3. Yellow: IN1
4. Purple: IN2

### Connections on Breadboard and more on PICAXE



On PICAXE (Left port)

1. Orange – Feedback
2. Red – Mode Switch / Potentiometer
3. Yellow – Power Switch

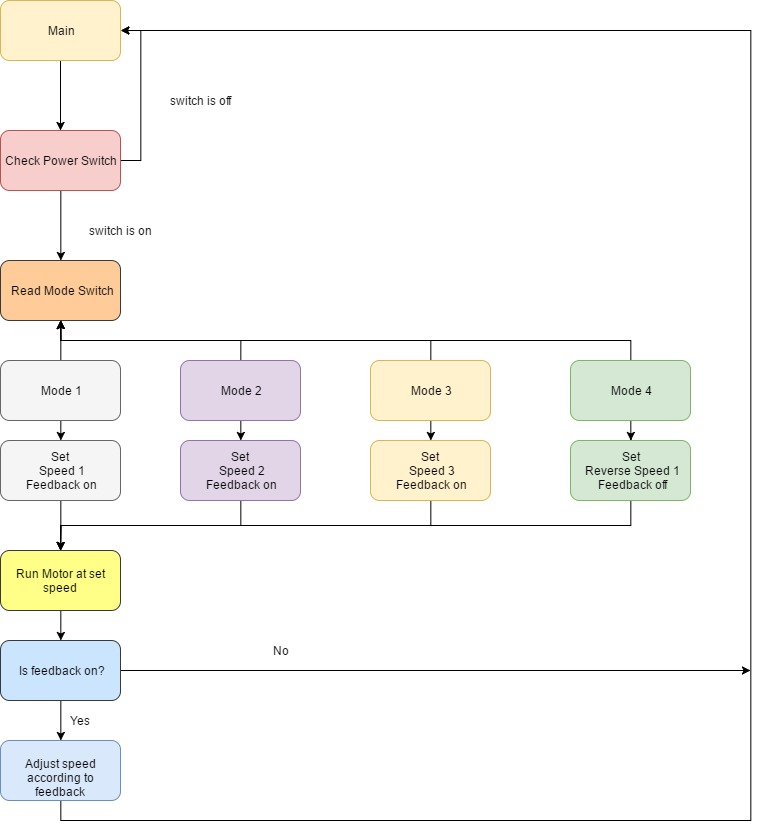
# Software Specifications

The main code is present in Appendix A, here we have included the base considerations and ideas which we made use of for writing the final code.

## Considerations

1. **Noise**Since the feedback provided by the motor is extremely noisy even after the inclusion of a low pass filter, we have setup a noise threshold level under which we do not activate the feedback loop and assume that the motor is running at nearly the expected speed.
2. **Max/Min speed**We do not want the feedback to reduce the duty cycle below a certain point if it thinks the speed is too much as that would make the motor stop or start running in the opposite direction. Also, we do not want the motor to run at a very high duty cycle as it may get damaged due to high voltage.   
   Therefore, while increasing or decreasing the duty cycle in feedback we reduce or increase it only till a pre-specified limit.

## Control Flow Diagram



## Feedback Analysis

We expected the output voltage obtained by the second motor to be a function of the duty cycle set on the first motor

And

The change in expected and obtained output voltage will relate to a change in duty cycle as

So, we would set the duty cycle as

The motor presented the following curve when we plotted the obtained feedback voltage with the supplied PWM duty cycle.

Even though the values followed the red curve more closely, we found that a piecewise linear model could be made use of for our specified speeds.

Following a piecewise linear model, we expect the output voltage to follow the following equation

This enables us to find the change in Duty required to be simply

We chose the following speeds for our closed loop circuit:

1. TODO
2. TODO
3. TODO

And for those values we found m as TODO

Since PICAXE does not allow floating point arithmetic, we do the following

# System performance

### Minimum Speed

We observed that the motors can’t run under a certain frequency for different loads, for that reason we had to modify the code so that the motors don’t stop while demonstrating.

### Feedback

The PWM output is altered by the PICAXE as expected from the feedback loop, under false feedback we observed that if the feedback is set to 0 the duty cycle is increased to the maximum permissible and if then the feedback is set to high (5V) then the duty cycle is decreased to minimum permissible.  
In real cases, it tries to stabilize the output near the expected output and ends up oscillating around it increasing and decreasing the duty cycle as it tries to match expected and obtained results.

### Noise

Noise is a major factor as even if the motors are running at the desired speed, they tend to send a spike of voltage which confuses the feedback loop.

### Overall

Overall the system matches our expected outputs, it runs in both clockwise and counter clockwise directions in accordance with the modes. The feedback loop tries to match the real output with the expected output as closely as it can.

# Appendix A (PICAXE Code)

#rem

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Setting variables and properties

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

#endrem

setfreq M32

symbol mainSwitch = b1

symbol modeSwitch = b2

symbol realVout = b3

symbol desiredFrequency = b4

symbol noiseLevel = b5

symbol feedbackFlag = b6

symbol expectedVout = b7

symbol diffVout = b8

symbol differenceFrequency = b9

symbol finalFrequency = b10

symbol mode = b11

symbol motorSetFlag = b12

symbol frequencyMin = b13

symbol frequencyMax = b14

mode=**0**

frequencyMin = **170**

frequencyMax = **220**

main:

readadc B.1,b1

readadc B.2,b2

#rem

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

If switch is off

set motor pwm at 50% => motor is off

continue loop

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

#endrem

if mainSwitch<**50** then

hpwm **1**,**0**,**0**,**79**,**159**

mode=**0**

goto main

endif

gosub modeController

gosub motor

goto main

#rem

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

modeController

used for setting the mode to run the device in desired mode

sets feedbackflag to enable/disable feedback loopback mode

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

#endrem

modeController:

if modeSwitch < **60** then

if mode != **1** then

sertxd("Mode 1",cr,lf)

desiredFrequency = **181**

'desiredFrequency = 218

finalFrequency = desiredFrequency

mode = **1**

feedbackFlag = **1**

motorSetFlag = **0**

endif

else if modeSwitch < **130** then

if mode != **2** then

sertxd("Mode 2",cr,lf)

desiredFrequency = **178**

'desiredFrequency = 198

finalFrequency = desiredFrequency

mode = **2**

feedbackFlag = **1**

motorSetFlag = **0**

endif

else if modeSwitch < **220** then

if mode != **3** then

sertxd("Mode 3",cr,lf)

desiredFrequency = **173**

'desiredFrequency = 170

finalFrequency = desiredFrequency

mode = **3**

feedbackFlag = **1**

motorSetFlag = **0**

endif

else

if mode != **4** then

sertxd("Mode 4",cr,lf)

desiredFrequency = **140**

finalFrequency = desiredFrequency

mode = **4**

feedbackFlag = **0**

motorSetFlag = **0**

endif

endif

return

#rem

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

motor

runs the motor at desired frequency

ensures speed if feedbackflag is set

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

#endrem

motor:

if motorSetFlag = **0** then

sertxd("final frequency for motor is ",finalFrequency,cr,lf)

hpwm **1**,**0**,**0**,**79**,finalFrequency

motorSetFlag = **1**

if feedbackFlag=**1** then

gosub feedback

motorSetFlag = **0**

endif

endif

return

#rem

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

feedback

makes sure that the motor is running

at desired speed within noise level

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

#endrem

feedback:

readadc B.3,b3

noiseLevel = **3**

if mode = **1** then

expectedVout = **13**

else if mode = **2** then

expectedVout = **9**

else

expectedVout = **4**

endif

sertxd("realVout is ",realVout,cr,lf)

if realVout > expectedVout then

diffVout = realVout - expectedVout

sertxd("diffVout is ",diffVout,cr,lf)

if diffVout < noiseLevel then

return

else

differenceFrequency = **100** \* diffVout

differenceFrequency = differenceFrequency / **125**

finalFrequency = finalFrequency - differenceFrequency

if finalFrequency < frequencyMin then

finalFrequency = frequencyMin

endif

endif

else

sertxd("Diff vout is negative",cr,lf)

diffVout = expectedVout - realVout

sertxd("diffVout is ",diffVout,cr,lf)

if diffVout < noiseLevel then

return

else

differenceFrequency = **100** \* diffVout

differenceFrequency = differenceFrequency / **125**

finalFrequency = finalFrequency + differenceFrequency

if finalFrequency > frequencyMax then

finalFrequency = frequencyMax

endif

endif

endif

sertxd("desired frequency is ",desiredFrequency,cr,lf)

sertxd("diffFrequency is ",differenceFrequency,cr,lf)

sertxd("final frequency is ",finalFrequency,cr,lf)

return