Inverted Pendulum

Sudhir Kumar (13D070036)
Kanhaiya Kumar(13D070046)
Khushhall Chandra Mahajan(13D070064)

Guide: Prof. Debraj Chakraborty
Guide: Prof. Salabh Gupta
Guide: Prof. Madhu N. Belur

April 15, 2016

Contents

1 Abstract

Inverted Pendulum is a device with an arm fixed at a fulcrum point that has a single axis of freedom. The objective is to apply force or torque to this pivot point so that the arm balances on its joint. The controlled point may be actuated in either linear or rotary orientation. This experiment considers the rotary inverted pendulum problem where a DC motor with rotation axis perpendicular to the axis of the fulcrum is connected to the base. The inverted state of the pendulum is an unstable equilibrium. Small errors or noise can destabilize it. Feedback control is required to continuously compensate for the noise. This experiment uses two sets of optical encoder and HCTL-2022 decoder to provide angular position of Pendulum and the Arm as feedback. We use Arduino environment, which uses this feedback and models an LQR controller to generate differential PWM signal that outputs to a motor driver circuit. The motor at the base is thus controlled bi-directionally using PWM signals output by the motor driver circuit. The motor driver circuit is used to generate both power and electrical isolation to protect the Arduino. We also use Octave environment for the mathematical model of the control system.

-Authors

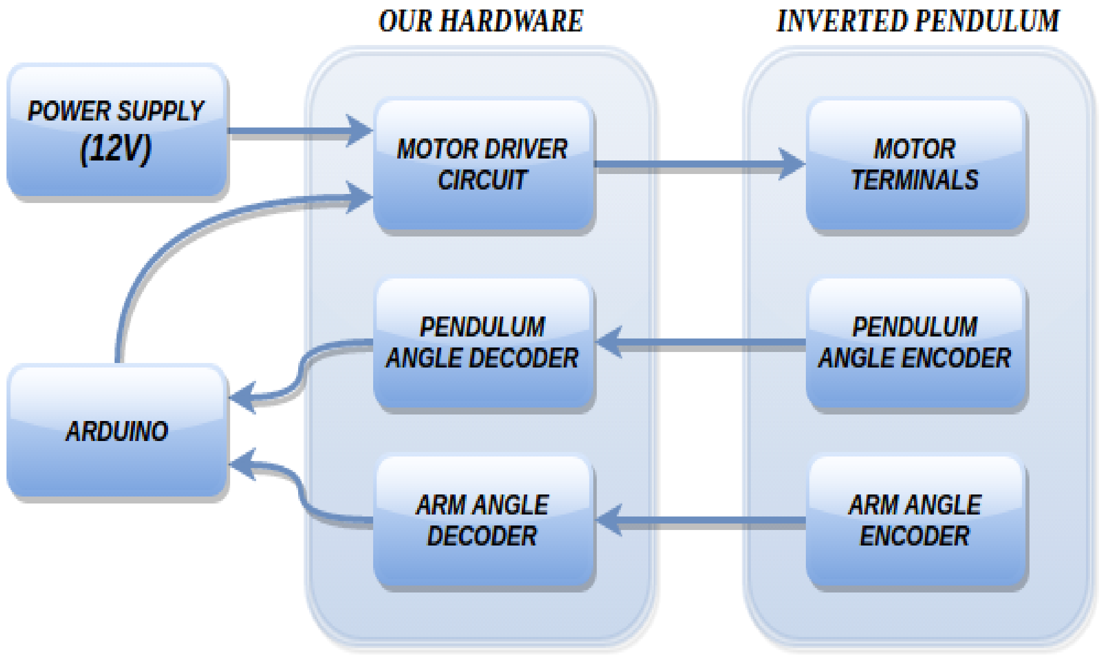
2 Aim

To stabilize the given pendulum in its inverted position by moving its hinge point using feedback controlled dc motor.

3 Introduction

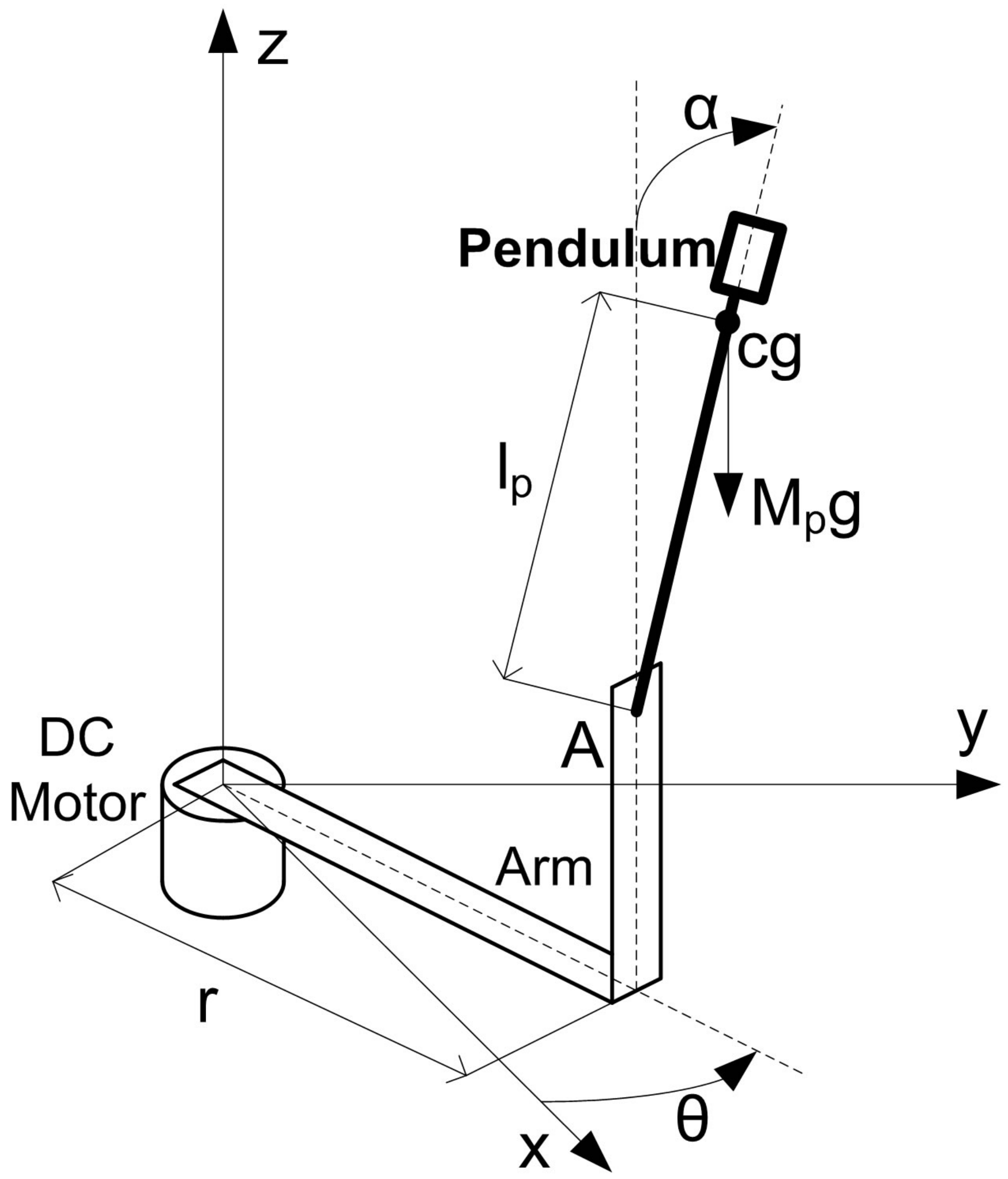
There is a pendulum, which is free to rotate in a plane about a hinge. The hinge itself can be rotated in the horizontal plane so that it can balance the pendulum in the inverted position. The movement of the hinge is controlled by a DC motor. There are two Quadrature encoders, one to measure the angular position of the pendulum link and another for the arm link. These Quadrature encoder converts angular displacement into digital pulses which are to be given to the HCTL-2022 decoder for counting the precise angle. Arduino program is written to interface with HCTL-2022 decoder to read both angles. The angles are processed using LQR controller in Arduino to generate a control signal. And this control signal is fed back to the motor using motor driver circuit to control the balance of the pendulum.

3.1 Block Diagram



4 Inverted Pendulum

4.1 The System



4.2 Mathematical Modelling of Pendulum

Symbol	***Description***	***Value***	***Unit***
M_p	Mass of the pendulum assembly (weight and link combined).	0.027	kg
l_p	Length of pendulum center of mass from pivot.	0.153	m
L_p	Total length of pendulum.	0.191	m
r	Length of arm pivot to pendulum pivot.	0.08260	m
J_m	Motor shaft moment of inertia.	3.00E-005	$kg{\cdot}m^2$
M_{arm}	Mass of arm.	0.028	kg
g	Gravitational acceleration constant.	9.810	m/s^2
J_{eq}	Equivalent moment of inertia about motor shaft pivot axis.	1.23E-004	$kg{\cdot}m^2$
J_p	Pendulum moment of inertia about its pivot axis.	1.10E-004	$kg{\cdot}m^2$
B_{eq}	Arm viscous damping.	0.000	N·m/(rad/s)
B_p	Pendulum viscous damping.	0.000	N·m/(rad/s)
R_m	Motor armature resistance.	3.30	Ω
K_t	Motor torque constant.	0.02797	N·m
K_m	Motor back-electromotive force constant.	0.02797	V/(rad/s)

The linear approximation of the system is given by:

$$\dot{X} = AX + BU$$

where,

$$X = \begin{bmatrix} \theta \\ \alpha \\ \dot{\theta} \\ \dot{\alpha} \end{bmatrix}$$

θ and α are as shown in the figure of section 4.1

State-Space Matrix	Expression
A	$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{rM_p^2 l_p^2 g}{J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2} & -\dfrac{K_t K_m (J_p + M_p l_p^2)}{(J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2) R_m} & 0 \\ 0 & -\dfrac{M_p l_p g (J_{eq} + M_p r^2)}{J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2} & \dfrac{M_p l_p K_t r K_m}{(J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2) R_m} & 0 \end{bmatrix}$
B	$\begin{bmatrix} 0 \\ 0 \\ \dfrac{K_t (J_p + M_p l_p^2)}{(J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2) R_m} \\ -\dfrac{M_p l_p K_t r}{(J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2) R_m} \end{bmatrix}$

We get A and B matrix by putting the values of variable from the table.

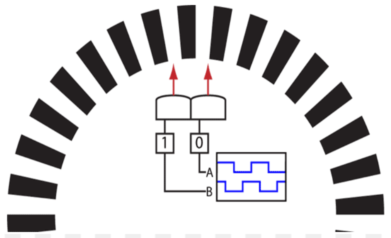
$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 123.98 & -1.577 & 0 \\ 0 & 111.62 & -0.7253 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 56.38 \\ 25.98 \end{bmatrix}$$

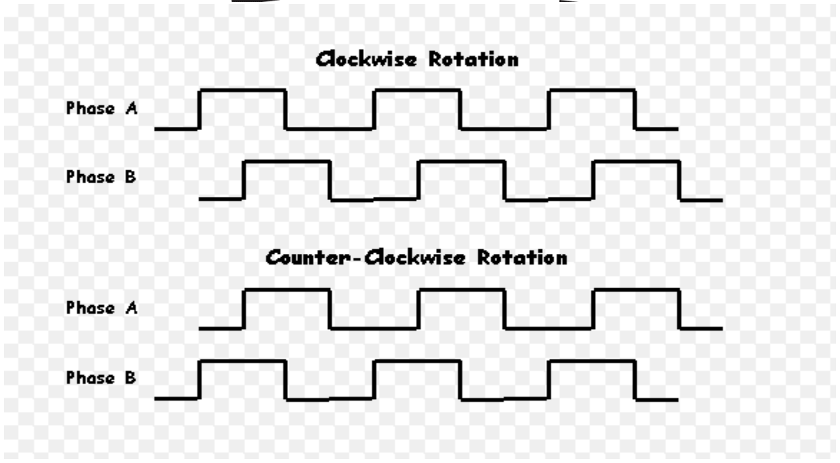
Note: The above expression assumes α from vertically downward, we have taken care of it and change the sign at appropriate place using eqn. 20 of page:
`http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeli`
We have designed LQR controller so, U =-B*K; In our experiment K= [-3.58 36.3 0 0] and the corresponding pole location is at

$$\begin{bmatrix} -0.8824 + 24.7224i \\ -0.8824 - 24.7224i \\ 0.0939 + 4.2383i \\ 0.0939 - 4.2383i \end{bmatrix}$$

Clearly, the poles are in negative half plane or nearer to imaginary axis. Therefore the system is stable. But due to imaginary part there will be some oscillation and the oscillation frequency is given by $\frac{24.7224}{2\pi} \simeq 4$. And our experimental result satisfies this value.

4.3 Angle Encoder

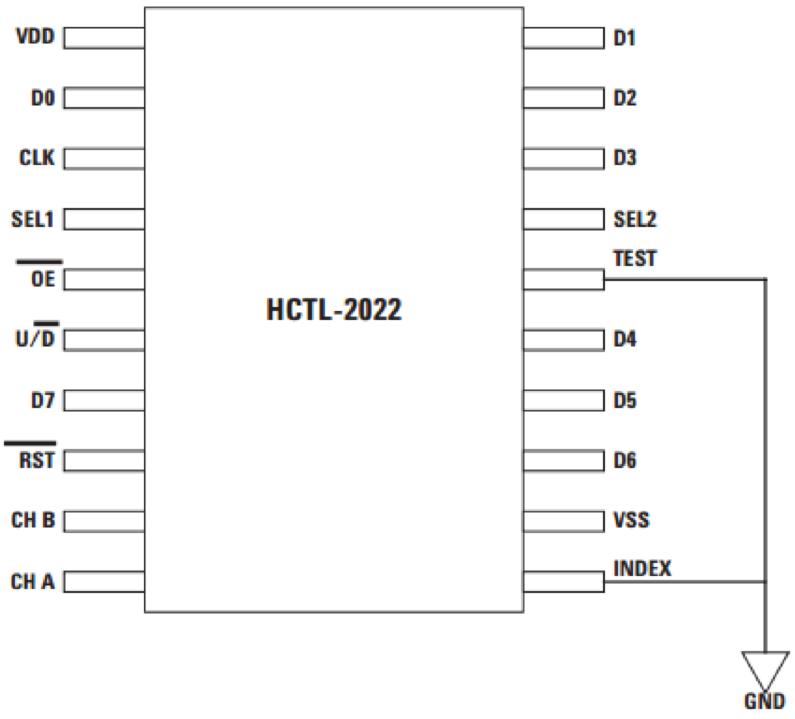




Quadrature Encoders are handy sensors that let you measure the speed and direction of a rotating shaft (or linear motion) and keep track of how far you have moved. A quadrature encoder normally has two outputs - Channel A and B - each of which will produce digital pulses when the thing they are measuring is in motion. These channels are coded 90 degrees out of phase, as indicated in the image, and this is the key design element that provides the quadrature encoder its functionality. There are 1024 slits in the disc so 1024 pulses will be generated in one complete rotation by each channels. Since the channels have 90 phase offset so we can measure the angle with quadruple resolution.
There are two Quadrature encoders, one to measure the angular position of the pendulum link and another for the arm link. These Quadrature encoder converts angular displacement into digital pulses which are to be given to the HCTL-2022 decoder for counting the precise angle.

5 Hardware

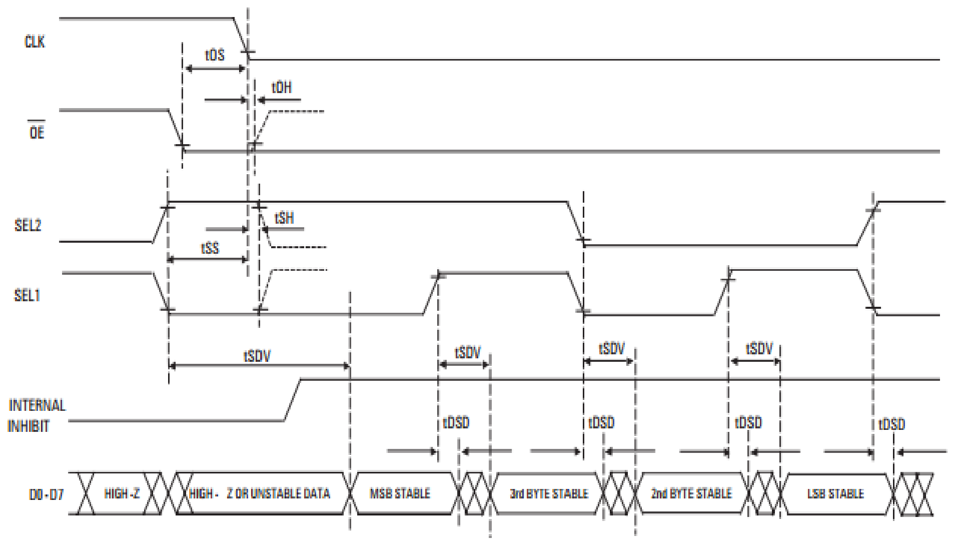
5.1 Angle Decoder



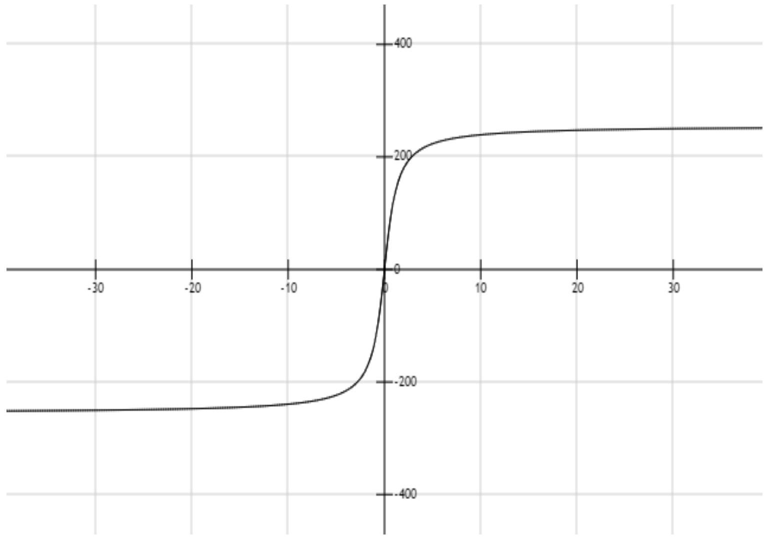
The HCTL-2022 is CMOS ICs that perform the quadrature decoder, counter, and bus interface function. The HCTL-2022 is designed to improve system performance in digital closed loop motion control systems and digital data input systems. It does this by shifting time intensive quadrature decoder functions to a cost effective hardware solution. The HCTL-2022 consists of a binary up/down state counter, and an 8-bit bus interface. The use of Schmitt-triggered CMOS inputs and input noise filters allows reliable operation in noisy environments. The HCTL-2022 contains 32-bit counter which will update only at the rising edge of clock. The clock pulse has to be provide from outside with a maximum up to 33 MHz (We are providing 25 kHz clock pulse from the arduino itself).

5.2 Interfacing with Arduino

We have used timer interrupt of 50 kHz frequency to generate the clock pulse of 25 kHz frequency (to be given to both the decoders). Initially reset pulse is sent to reset the current counter value of both decoders. OE, TEST, VSS and INDEX are grounded.



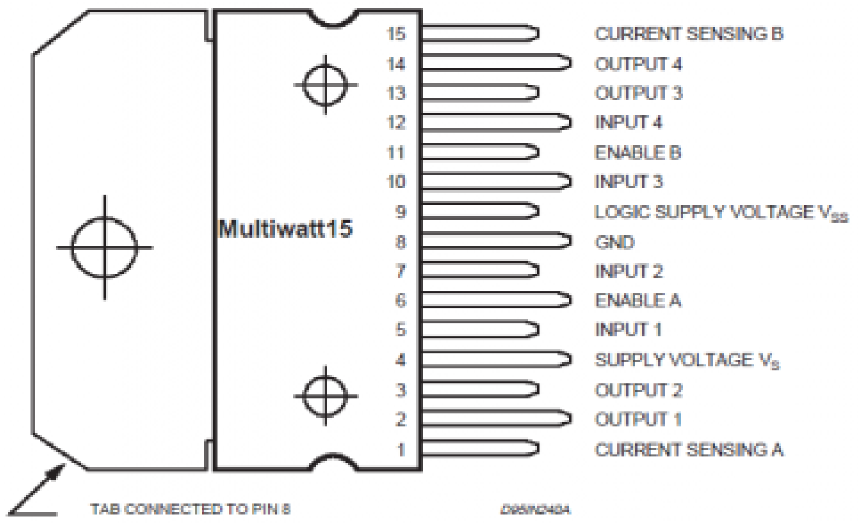
Based on the above Timing Diagram we made the functions alphadecoder() and thetadecoder() which will read the angle values from the decoders (Note: Instead of digitalread() and digitalwrite() we have used direct read or write technique to make the system fast). The main loop takes around 0.8 msec to complete one cycle. So the above function gets called at every 0.8 msec. So this time is actually the time it takes to update our control signal. The control signal is generated using LQR controller design (as described earlier). Since the response of dc motor is not linear with the voltage applied across it, there is a lower threshold of duty cycle below which the motor doesn't move. So, we have passed this control signal through a function which will boost its value when it is lower and will keep around constant when it is higher (shown in the figure). By tuning the parameter k, we can control the boosting.



Now the control signal is converted into two PWM signals (differential signal) using analogwrite() command of Arduino. We have set the frequency of PWM to 3.9 kHz (Note: this frequency must be greater than our control signal updating frequency $\sim$ 1.25 kHz) This differential PWM signal is given to motor driver circuit.

5.3 Motor Driver Circuit

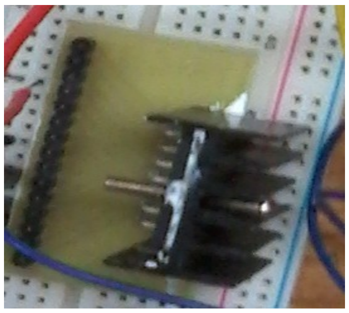
According to the specifications of base motor of the setup of inverted pendulum we need a motor driver circuit which can provide currents upto 1A. For the motor driver we used L298N with the connections shown as below in the diagram.



One important point that must be kept in mind is that since the motor driver draws high current, connections on breadboard may melt down the breadboard so it is strongly suggested to do basic routing of the circuit on pcb.

5.3.1 Heat Sink

L298N gets heated up on operating for 10-15 minutes continuously because of the current mentioned above. So heat sink must be fitted along with this IC for heat dissipation. Heat dissipation gets smooth and faster when we apply thermal paste before fitting with ic. Thermal paste applied layer should be kept very thin otherwise applying thick thermal paste may get detrimental.



6 PCB

6.1 PCB Design

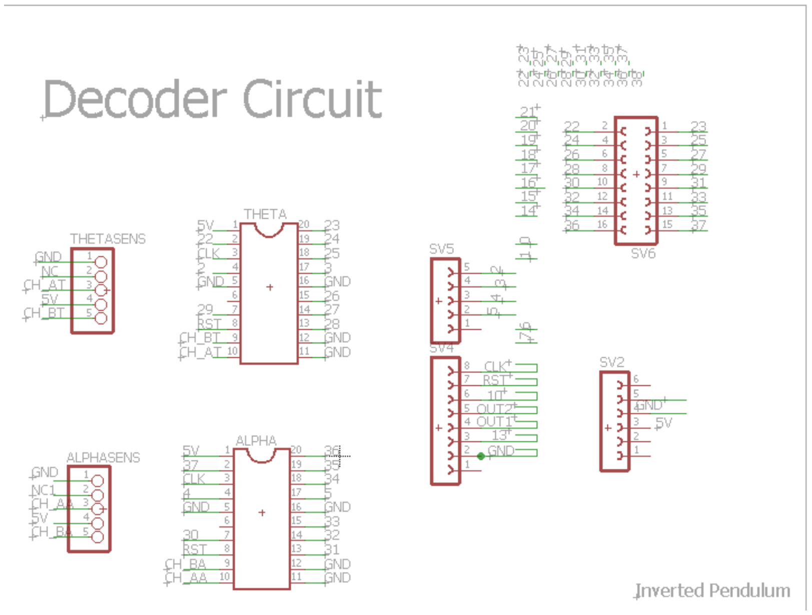
Once our design was finalized and our model was working fine it was needed to give a complete professional look to our efforts. Our whole circuit has two parts:

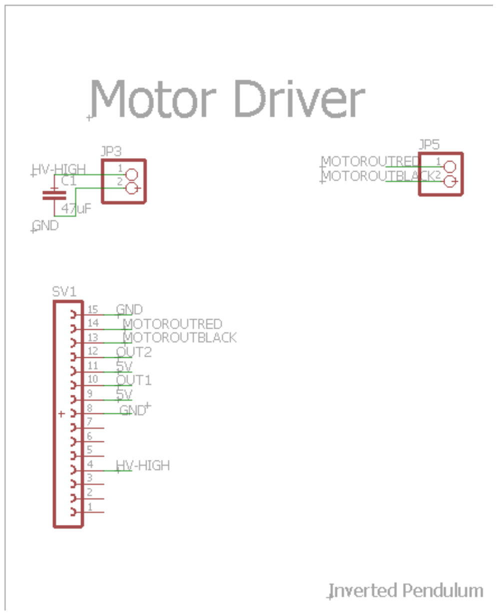
1. Decoder circuit(HCTL 2022)

2. Motor driver circuit

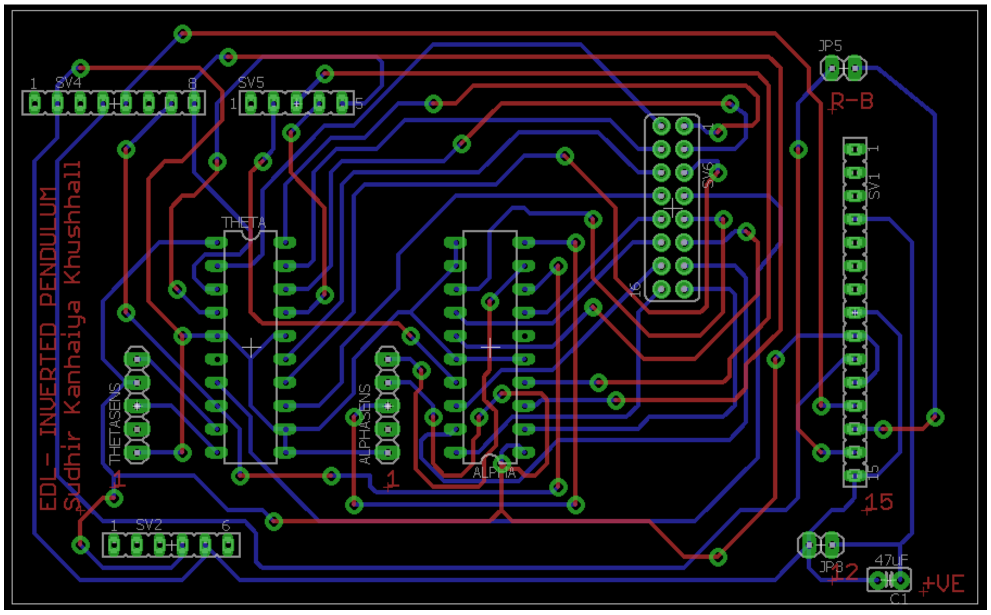
We tested the complete circuit on breadboard and since decoder have 20 pins and we are using two decoders to decode both the sensors reading, approximately 40 jumpers are needed. Along with it our motor driver circuit too have approx 6-8 connections. In this way the whole setup looks a mess due to such large no of jumpers. In any case by accident, if any of the jumper's connection gets loose the pendulum stability will not be achieved. Also motor driver circuit may require current upto 1A during the working. Then due to large current heating breadboard may melt down. So to use this design as a complete usable product we moved towards designing the PCB to get rid of this whole mess. And since we were using Arduino MEGA 2560 we decided to make a shield which could fit on this Arduino directly on the top.

6.2 PCB Schematic

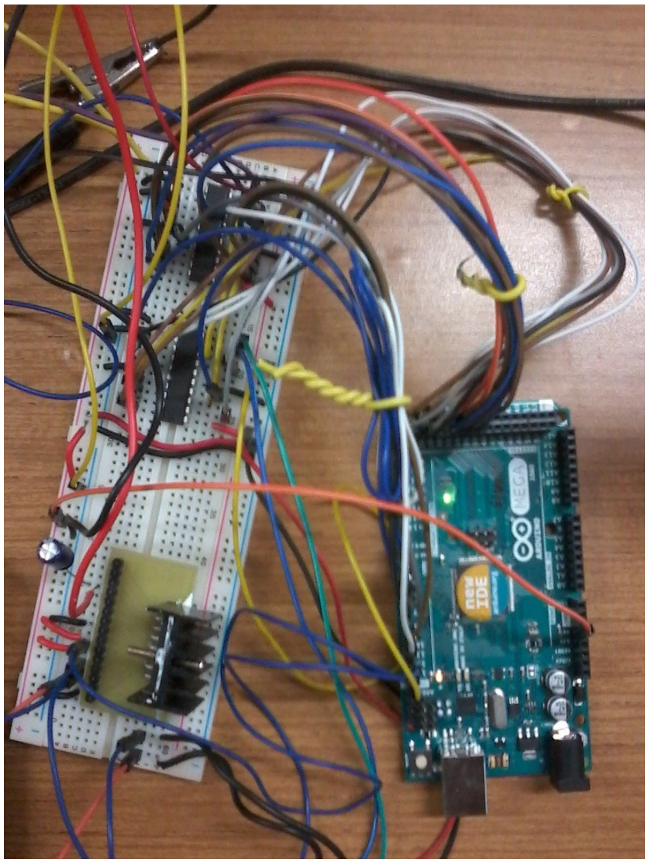




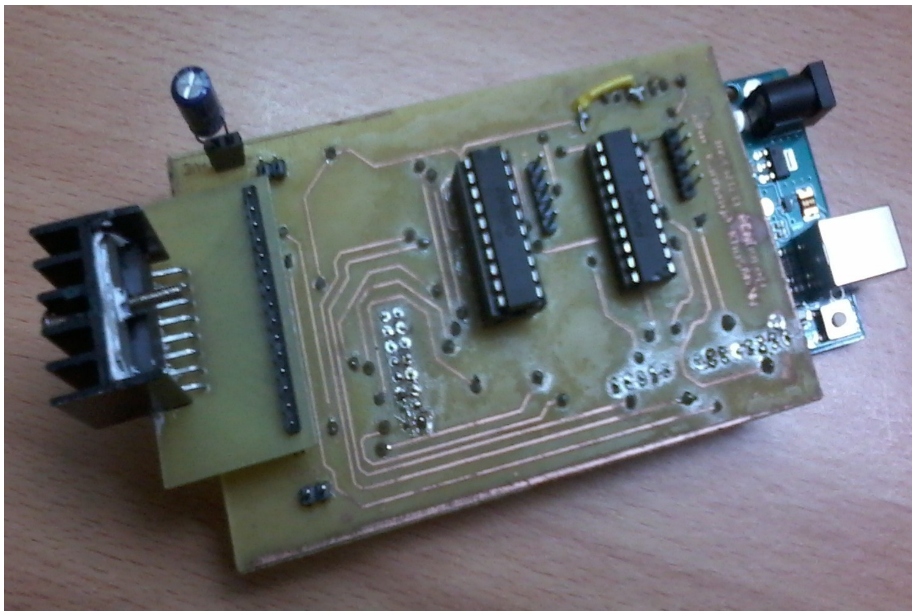
6.3 PCB BRD



6.4 Before PCB design



6.5 After PCB design



7 Results and Future improvements

- The pendulum gets stable itself for around 2-3 mins. It is taking around 0.5-0.8 Ampere current in stabilized position
- Due to high current consumption by the motor, the heat sink starts heating. So the system could not be stablize for a long duration.
- There is also some vibration in the stablized position (which has been discussed in section 4.2). Here we have not taken the contribution of derivative of alpha and theta terms in our feedback controller design because our system has too much of vibration which may enhance the noise if we take directly derivative of alpha and theta terms.
- This issue can be resolved by observer design model of estimating the state variables.

$$\hat{x}(k+1) = (A - BK)\,\hat{x}(k) + L\,(y(k) - \hat{y}(k))$$
$$\hat{y}(k) = (C - DK)\,\hat{x}(k)$$

- By estimating the state variables correctly we can reduce the vibration in our system to a greater extent and this will also lead to lower current requirement in the steady state. That will effectively increase the total time of stablization because the heat sink in that case heat sink will heat less.

8 References

- QNET Inverted Pendulum Laboratory Manual
- `http://file.scirp.org/Html/5-7600297_42071.htm`
- `http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling`
- `http://www.creative-robotics.com/quadrature-intro`
- `http://www.mindspring.com/~tom2000/Delphi/Codewheel.html`
- Avago Technologies