Chapter 3

Inverted Pendulum Control

3.1 Problem Statement:

Design a linear Quadratic Regulator balance controller for the Inverted Pendulum. After manually initializing the pendulum in the upright vertical position, the balance controller should move the rotary arm to keep the pendulum in this upright position. Moreover it should be capable of balancing itself, even if minor external disturbances are given.

3.2 Inverted Pendulum setup

The set up consists of a 12 Volt DC motor that is coupled with an encoder and is mounted vertically in the metal chamber. The L-shaped arm is connected to the motor shaft that can rotate 360 degrees. At the end of the arm, there is a suspended pendulum attached.

To measure the pendulum angle and the rotation angle of L-shaped arm, two encoders were used. Two decoders were used to covert these encoded value to binary values before passing it to the micro-controller. A 12 V DC supply was given to the motor through motor driver circuit. The VCC of the two encoders and the VDD of the two decoders were connected to the 5V internal supply of the microcontroller. The circuit diagram of the setup is shown in figure 3.1.

Decoder used: HCTL-2022 Avago Technologies
Motor Driver: L298N
Microcontroller: Arduino Mega 2560

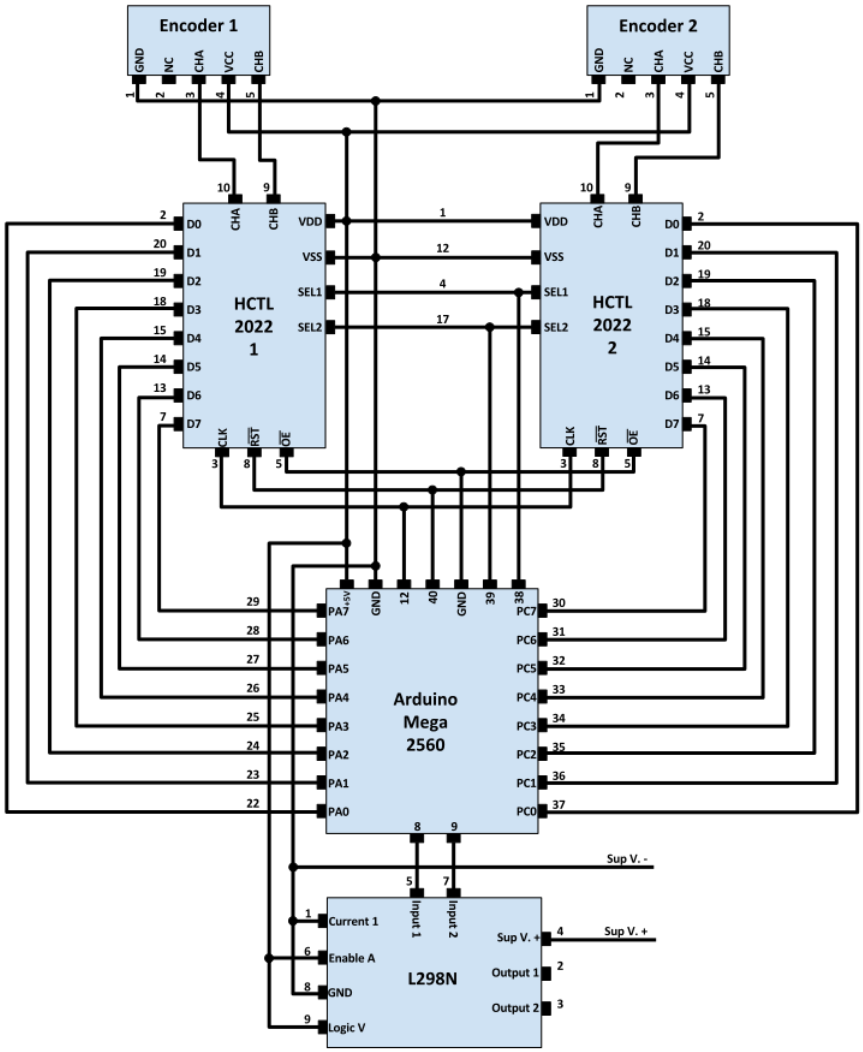

Figure 3.1: Schematic circuit diagram

3.3 Model Parameters

Symbol	Description	Value	Unit
M_p	Mass of the pendulum assembly (weight and link combined).	0.027	kg
I_p	Length of pendulum center of mass from pivot.	0.153	m
r	Length of arm pivot to pendulum pivot.	0.08260	m
J_m	Motor shaft moment of inertia.	3.00e-005	kg-m^2
M_arm	Mass of arm.	0028	m
g	Gravitational acceleration constant.	9.810	m/s^2
J_eq	Equivalent moment of inertia about motor shaft pivot axis.	1.23e-004	kg-m^2
J_p	Pendulum moment of inertia about its pivot axis.	1.10e-004	kg-m^2
B_eq	Arm viscous damping.	0.00	N-m/(rad/s)
B_p	Pendulum viscous damping.	0.00	N-m/(rad/s)
R_m	Motor armature resistance.	3.30	Ω
K_t	Motor torque constant.	0.2797	N-m
K_m	Motor back-electromotive force constant.	0.2797	V/(rad/s)

Table 3.1: Model Paarameters

3.4 State Space Model

The inverted pendulum has two degree of freedom. The arm rotates about the vertical axis and its angle is denoted by the symbol θ while the pendulum attached to the arm rotates about its pivot (horizontal axis) and its angle is called α. The shaft of the DC motor is connected to the arm pivot and the input voltage of the motor is the control variable.

In the linearzed model the states are $x_1 = \theta$, $x_2 = \alpha$, $x_3 = \dot{\theta}$, $x_4 = \dot{\alpha}$.

$$x = [x_1, x_2, x_3, x_4]^T$$

The linear state-space representation of the Inverted Pendulum is

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{3.1}$$

where $u = V_m$ and A,B,C,D matrices are

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{rM_p^2 l_p^2 g}{J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2} & \frac{-K_t K_m (J_p + M_p l_p^2)}{(J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2) R_m} & 0 \\ 0 & \frac{M_p l_p g (J_{eq} + M_p r^2)}{J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2} & \frac{-M_p l_p K_t r K_m}{J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2) R_m} & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ \frac{K_t (J_p + M_p l_p^2)}{J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2) R_m} \\ \frac{M_p l_p k_t r}{J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2) R_m} \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

3.5 LQR Control design

For a given plant model

$$\dot{x(t)} = Ax(t) + Bu(t)$$

linear quadratic regulator problem is to find a control input u that minimizes the cost function

$$J = \int_0^\infty x^T(t) Q x(t) + u^T(t) R u(t) dt$$

where Q is an $n \times n$ positive semidefinite weighing matrix and R is an $r \times r$ positive definite symmetric matrix. That is, find a control gain K in the state feedback control law

$$u = Kx$$

such that the quadratic cost function J is minimized.
The Q and R matrices affects the optimal control gain that is generated to minimize J. The closed-loop control performance is affected by changing the Q and R weighing matrices. The controller computes a voltage V_m depending on the position and velocity of the arm and pendulum angles. The performance index J can be interpreted as an energy function, so that making it small keeps small the total energy of the closed-loop system. Both the state $x(t)$ and the control input $u(t)$ are weighted in J, so that if J is small, then neither $x(t)$ nor $u(t)$ can be too large. Note that if J is minimized, then it is certainly finite, and since it is an infinite integral of $x(t)$, this implies that $x(t)$ goes to zero as t goes to infinity. This in turn guarantees that the closed loop system will be stable.

3.6 Code implementation

The code was implemented in Arduino MEGA 2560. It acts as the controller for the system. It takes the values from the decoder, computes the optimum input u by $u = Kx$ and gives the control signal to the motor driver for balancing the pendulum. The K matrix was calculated through Matlab.

3.6.1 Arduino

The Arduino Mega is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs) and 16 analog inputs. The pin connections of Arduino MEGA 2560 for our setup are shown in the table 3.2.

PIN number	Connected to	Mode
8-9	connected to DC motor	Output mode
12	CLK of Decoder 1 and Decoder 2	Output mode
22 - 29 (PORT A)	Input from Decoder 1	Input mode
30 - 37 (PORT C)	Input from Decoder 2	Input mode
38	SEL1 of decoder 1 and decoder 2	Output mode
39	SEL2 of decoder 1 and decoder 2	Output mode
40	$\overline{\text{RST}}$ of decoder 1 and decoder 2	Output mode

Table 3.2: Arduino Pin connections

In the initial setup, we set pin 40 High and we set the output clock frequency to 16KHz.There are 6 timers in Arduino Mega board.Timer 1, Timer 3, Timer 4 and Timer 5 are 16 bit timers where as the Timer0 and Timer 2 are 8 bit Timers. We can change the Timer behaviour through the timer register TCCRnB, where n represents the Timer. Here we selected Timer1, and set the prescale to 1. So the timer frequency is 31KHz. Timer 1 controls pin 12, 11 and they are also PWM pins. So we can set the output clock frequency by setting a pin to analog value 128.

Listing 3.1: clock setup

```
TCCR1B=0x01; //set timer 1 frequency to 31kHz
analogWrite(12,128);
```

In the next part we take the inputs from the decoder, and calibrate it to 360 degree movement of the arm and the pendulum. Now, the HCTL-2022 decoder can send 8 bits (1 Byte) data at time. Depending on SEL1 and SEL2 a byte from the 4 byte data would be sent. In this case we need only two bytes of data from both the decoders. The code

for calibration of decoder value for one decoder is shown in Listing 3.2. For one full 360

SEL1	SEL2	Byte selected
0	1	MSB
1	1	2nd Byte
0	0	3rd Byte
1	0	LSB

Table 3.3: Byte selection via select pin

degree rotation of the arm and the pendulum, the decoder values are more than 256 but less than 65536. So we just need the last two bytes of data from the decoder. Here first we get stable data for 3rd Byte and the least significant byte from decoder 1 using functions *getMSBA()* and *getLSBA()*. Then we merge the two bytes to get a single 16 bit data. This is done by function *mergeFunc()*. *count_en1* converts the 16 bit data to an integer value. Each rotation gives a count value of 2000 approx. so for converting it to corresponding degree value, we multiply *count_en1* by $\frac{360}{2000} = 0.18$. Same thing is done for calibrating the data given by the encoder 2. This will ensure that all the values of α and θ given by *en1* and *en2* are in degrees.

Listing 3.2: Calibrate decoder value

```
byte getMSBA(){
/*Get stable data for the most significant byte of countData*/
  byte MSBold = PINA;
  byte MSBnew = PINA;
  if (MSBnew == MSBold){
    byte MSBresult = MSBnew;
    return MSBresult;
  }
  else getMSBA();

byte getLSBA(){
/*Get stable data for the least significant byte of countData*/
  byte LSBold = PINA;
  byte LSBnew = PINA;
  if (LSBnew == LSBold){
    byte LSBresult = LSBnew;
    return LSBresult;
  }
  else getLSBA();
```

```
}

long mergeFunc(byte MSBresult,  byte LSBresult){
  long tempVar = 0;
  tempVar |= ((long) MSBresult << 8) |((long) LSBresult << 0);
  countData = tempVar;
  return countData;
}
digitalWrite(38, LOW);
digitalWrite(39, LOW); // SEL1 = 0 and SEL2 = 0
byte enc1_MSB = getMSBA();

digitalWrite(38, HIGH);
digitalWrite(39, LOW);
byte enc1_LSB = getLSBA();

count_en1 = int(mergeFunc(enc1_MSB,enc1_LSB));

en1 = 0.18*count_en1;
```

After θ and α are calculated, we calculate $\dot{\theta}$ and $\dot{\alpha}$. The K matrix is computed in Matlab by LQR method. Then input to motor is calculated as $u = Kx$.

Listing 3.3: input calculation

```
float u = k1*theta + k2*alpha + k3*d_theta +k4*d_alpha;
```

If we see the values of k in the final code we will find that value of $k2$ is significantly higher than other values. This represents that a very minor change in α will be corrected faster by increasing the motor voltage more than in other cases. The Final Arduino code is shown in Listing 3.4.

Listing 3.4: Final Arduino Code

```
/*
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Reading decoder HCTL-2022 to count pulses from
an encoder using Arduino Mega 2560
Written by Ayan Sengupta and Sarthak Sharma
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Arduino Pin       HCTL-2022 function
-----------------------------------
```

```
DIG 40              /RST (active low)
DIG 38              SEL1
DIG 39              SEL2
DIG 22->29          Data pins D0->D7
DIG 30->37          Data pins D0->D7 for 2nd Decoder
PIN12               CLK
8-9                 motor inputs
*/

volatile long countData = 0;
volatile long count_en1 = 0;
volatile long count_en2 = 0;
long int en1,en2;
int t1,t2,dt;
float prev_theta,prev_alpha;

void setup() {
  t1=0;
  Serial.begin(9600);
  pinMode(12, OUTPUT);
  pinMode(38,OUTPUT);
  pinMode(39,OUTPUT);
  pinMode(40, OUTPUT);
  digitalWrite(40, LOW); // /RST resets the internal counter between ru
  delay(10);
  digitalWrite(40, HIGH); // Stay high for rest of the run
  TCCR1B=0x01; //set timer 1 frequency to 31kHz
  analogWrite(12,128);


  // Set all pins in PORTA (digital pins 22->38 on the Mega) as input p
  for(int i = 22; i<38; i++) {
    pinMode(i, INPUT);
  }
  for(int j = 8 ;j<10; j++) {
    pinMode(j, OUTPUT);
    digitalWrite(j, LOW);
    }
}
byte getMSBA(){
```

```
  byte MSBold = PINA;
  byte MSBnew = PINA;
  if (MSBnew == MSBold){
    byte MSBresult = MSBnew;
    return MSBresult;
  }
  else getMSBA();
}
byte getMSBC(){
  byte MSBold = PINC;
  byte MSBnew = PINC;
  if (MSBnew == MSBold){
    byte MSBresult = MSBnew;
    return MSBresult;
  }
  else getMSBC();
}
byte getLSBA(){
/*Get stable data for the least significant byte of countData*/
  byte LSBold = PINA;
  byte LSBnew = PINA;
  if (LSBnew == LSBold){
    byte LSBresult = LSBnew;
    return LSBresult;
  }
  else getLSBA();
}
byte getLSBC(){
/*Get stable data for the least significant byte of countData*/
  byte LSBold = PINC;
  byte LSBnew = PINC;
  if (LSBnew == LSBold){
    byte LSBresult = LSBnew;
    return LSBresult;
  }
  else getLSBC();
}


long mergeFunc(byte MSBresult,  byte LSBresult){
```

```
  long tempVar = 0;
  tempVar |= ((long) MSBresult << 8) |((long) LSBresult << 0);
  countData = tempVar;
  return countData;
}
void loop() {

  digitalWrite(38, LOW);
  digitalWrite(39, LOW); // SEL1 = 0 and SEL2 = 1

  byte enc1_MSB = getMSBA();
  byte enc2_MSB = getMSBC();

  digitalWrite(38, HIGH);
  digitalWrite(39, LOW);

  byte enc1_LSB = getLSBA();
  byte enc2_LSB = getLSBC();

  count_en1 = int(mergeFunc(enc1_MSB,enc1_LSB));
  count_en2 = int(mergeFunc(enc2_MSB,enc2_LSB));

  en1 = 0.18*count_en1;
  en2 = 0.18*count_en2;
  t2=millis();

  dt=t2-t1;

  float theta=en1;
  float alpha=en2;
  float d_theta=((theta-prev_theta)/dt)*1000;
  float d_alpha=((alpha-prev_alpha)/dt)*1000;

  float k1 = 9.4;
  float k2 = 98.5;
  float k3=  -4.1;
  float k4= 13;

  float u = k1*theta + k2*alpha + k3*d_theta +k4*d_alpha;
  u = 0.3*u;
```

```
  if (u>0){
    if (u>255){
      u =255;
    }
  analogWrite(8,u);
  analogWrite(9,0);
  }
  if (u<0){
    if(u<-255){
      u = -255;
    }
  analogWrite(8,0);
  analogWrite(9,abs(u));
  }
prev_theta=theta;
prev_alpha=alpha;
t1=t2;
delay(20);
  Serial.print(en1);
  Serial.print("\t");
  Serial.print(en2);
  Serial.print("\t");
  Serial.println(t2);

  }
```

3.6.2 Matlab

After putting all the system parameter values in A and B matrices we get

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 123.979 & -1.577 & 0 \\ 0 & 111.623 & -0.725 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ 56.389 \\ 25.930 \end{bmatrix}$$

In matlab we can directly compute the optimal gain matrix K. For calculating the K matrix, we need matrix A,B,Q and R. We have taken R as an 1 as we do not want to put

any penalty on input, i.e. on input voltage. The Q matrix we used is

$$Q = \begin{bmatrix} 90 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} R = 1$$

Listing 3.5: Calculate matrix K

```
A = [0,0,1,0; 0,0,0,1; 0,123.979,-1.577,0; 0,111.623,-0.725,0]
B = [0; 0; 56.389; 25.389]
Q = [90,0,0,0; 0,5,0,0; 0,0,2,0; 0,0,0,1;]
R = 1
K = lqr(A,B,Q,R)
% output:
K = [ -9.487   98.198   -4.151   13.021]
```

3.7 Results

The following results and graphs are obtained with Q and R matrices as

$$Q = \begin{bmatrix} 90 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} R = 1$$

The variation of angle α and θ with respect to time are shown below. It shows the variation when external disturbances were given and the controller managed to keep the pendulum in the upright position. For the given disturbance the maximum deviation of angle α is 10 degree and the maximum devation of angle θ is around 50 degrees.

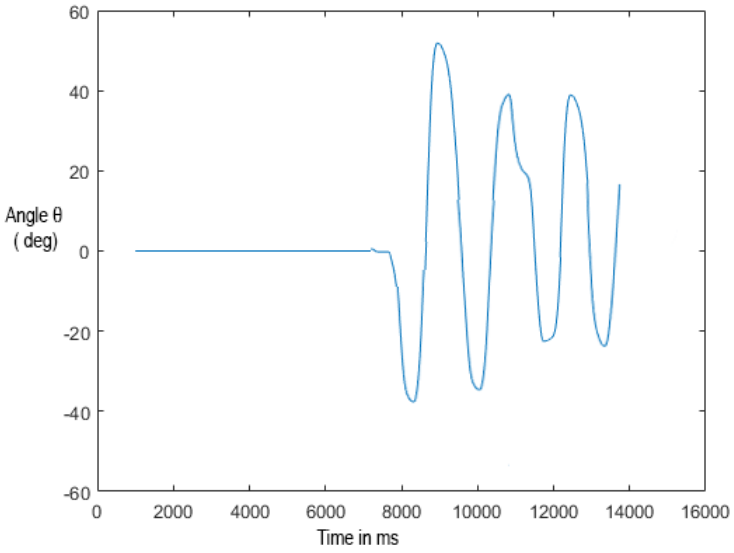

Figure 3.2: Variation of angle θ with time

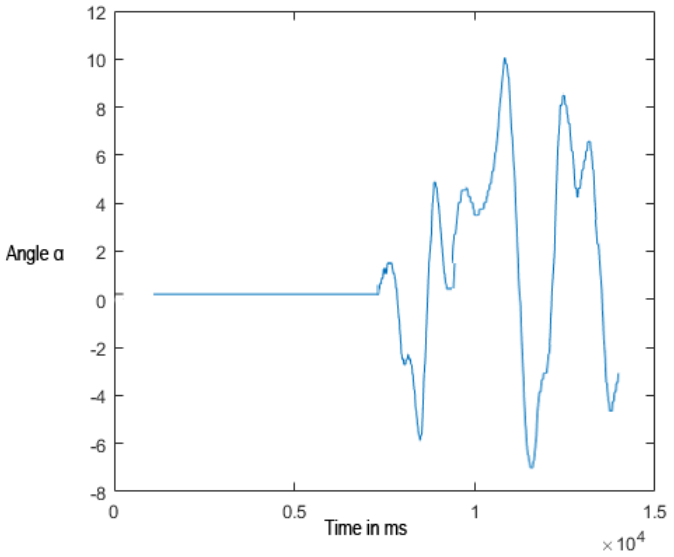

Figure 3.3: Variation of angle α with time

3.8 Challenges

The experiment took three weeks to complete. The challenges we faced during performing the experiment are discussed below.

- Tuning of Q matrix was a key part of this experiment. It took a lot of time to modify the Q matrix which gives desirable result. Then minor modifications were needed to make the controller more resistant to disturbances. Tweaking delay time was also crucial to get the perfect balancing pendulum.
- Another major observation is that, The K matrix used for one setup may not work for other setups. If we change the hardware setup then the system parameters get altered. Though the change might be small, still it requires another round of modifying the Q matrix and testing.