Inverted Pendulum

Introduction

An inverted pendulum is a pendulum in inverted position of in other words, it is a pendulum with its centre of gravity above its pivot point. Unlike a normal pendulum, inverted pendulum is unstable and would fall down without any external action of force. It needs constant external force to keep it in upright position. This project focuses on inverted pendulum with only one degree of freedom i.e. pivoted.

The main objective of this project was to learn the basic elements of a mechatronics system and understand how the different elements integrate and operate together to form a complete system. This project does not go into details of control systems theory and mathematical modelling and is simple for most people with basic knowledge of electronics to understand.

Objectives

At the beginning of the project certain objectives were decided to evaluate at the end the success or failure of the overall project. Following are the primary and secondary objectives,

Primary: Make an inverted pendulum which can balance on its own once it is brought to the inverted position manually.
Secondary: Swing up control of the inverted pendulum so that it can swing up from normal position to inverted position and then balance itself.

Theory

There are several different algorithms that can be used to balance an inverted pendulum. In this project PID (proportional-integral-derivative) type of control is used to balance the inverted pendulum. PID is very widely known, and there is plenty of information available on the web about it. Besides it is fairly simple to understand how the PID control works.

If we do a root locus analysis from the transfer function, we can see that only proportional and derivative control is required to balance an inverted pendulum. The transfer function for an inverted pendulum using proportional and derivative control is,

\[ H(s) = \frac{1}{s^2 + s\left(\frac{K_2}{L}\right) - \frac{g}{L} + \frac{K_1}{L}} \]

Where,
- \( s \) = poles of the equation,
- \( L \) = length of the pendulum,
- \( g \) = acceleration due to gravity,
- \( K_1, K_2 \) = Constants

Therefore the poles of the system can be evaluated as,

\[ s = \frac{-K_2}{2L} \pm \sqrt{\left(\frac{K_2}{2L}\right)^2 - \left(\frac{K_1 - g}{L}\right)} \]

By plotting the poles on the 'S' place we find out that the system is stable.

The theory described above can be found in more detail in the references section of this report along with explanatory videos. The theory only proves the stability of the pendulum however the stability of the cart should also be taken into account. Further information on this is in the control algorithm section of the report.

Parts required
There are several different configurations of an inverted pendulum. They are,

- Furuta pendulum
- Inertia wheel pendulum
- Inverted pendulum on a cart

For this project, the time available for building was one of the major challenges and one of the ideas for building was to use a printer mechanism for the linear motion and attach a pendulum to it. This would help considerably reduce the building and troubleshooting time with the mechanical system. Besides, the printer mechanism is well built and has a sensor for tracking linear motion and lastly it would also reduce the overall cost of the project. Considering the above advantages, it was decided to build an inverted pendulum on a cart.

Mechanical

The mechanical parts required for this project were,

- Linear actuator preferably using a DC motor since it is easy to control using a microcontroller.
- Bearing for mounting the inverted pendulum at pivot point.

Electronics

The electronic parts required were,

- Power supply for driving the motor and microcontroller.
- Microcontroller for controlling the actuator: DC motor.
- A motor driver circuit.
- Angle sensor for measuring and tracking the position of the inverted pendulum.
Cart with belt drive, linear encoder and dc motor
Rotary magnetic sensor for angle measurement
Since the linear actuator was used from a printer, and it was difficult to determine the parameters of the system for modelling, a virtual system model was not made. The method used was to determine the PID parameters manually by trial and error. By this method, even though the system may not operate optimally, it can provide satisfactory performance.

As the theory explains, PD control can balance an inverted pendulum, however it does not say anything about the stability of the cart. This was the main problem in the project since the printer mechanism travel was limited to ±18 cms. Hence, the final control algorithm had one more additional PI control along with the PD control such that PI loop calculated the set point for the angle sensor based on the cart position relative to the centre. The cycle times for the PI and PD loops were 10ms and 1ms respectively. The block diagram explaining the control algorithm is shown below.
Controller block diagram

As seen from the block diagram as soon as the controller is started with cart at centre and pendulum in inverted position, the linear encoder starts tracking the cart position and PI control calculates the set point for the angle based on that. Using this set point, the PD controller controls the DC motor driving the belt drive which moves the cart.

Problems encountered

Following are some of the problems encountered during the project.

- The PCBs made for angle sensor were etched wrong and mirrored by mistake since it was my first time making PCBs.
- A separate motor driver had to be made for the 40V power supply from the printer.
- Stabilizing the cart was a major challenge with the limited travel length.
- The wires for DC motor and analog angle sensor were close to each other, due to this high frequency PWM induced noise in the angle sensor.
- Swing up control was much more complex than expected and was not accomplished in the project.

Budget

<table>
<thead>
<tr>
<th>Components</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino UNO (Clone)</td>
<td>12€ (Free shipping)</td>
</tr>
<tr>
<td>Rotary Magnetic Sensor (Austria microsystems)</td>
<td>Free samples</td>
</tr>
<tr>
<td>Pendulum</td>
<td>Free (From scrap aluminium)</td>
</tr>
<tr>
<td>Cart, DC motor, Linear encoder</td>
<td>Free (From printer)</td>
</tr>
<tr>
<td>DC motor driver (L298n – STMicroelectronics)</td>
<td>5.5€</td>
</tr>
<tr>
<td>Power supply</td>
<td>Free (From printer)</td>
</tr>
<tr>
<td>PCBs</td>
<td>3€</td>
</tr>
<tr>
<td>Prototype PCBs</td>
<td>3€</td>
</tr>
<tr>
<td>Rotary angle sensor housing</td>
<td>Free (From scrap aluminium)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>23.5€</strong></td>
</tr>
</tbody>
</table>

Conclusion

This project was a good platform for an introduction to mechatronics and good hands on work. In this project offered all kinds of different work like CAD, machining, assembling, making PCBs, soldering, troubleshooting and programming. Considering the objectives that were set and the final result of the project, this project was a success. A video of the inverted pendulum in action can be found using the youtube link below,

Video: [https://www.youtube.com/watch?v=eklh7HAMa8s](https://www.youtube.com/watch?v=eklh7HAMa8s)

References and links

- The C Programming Language - Brian Kernighan and Dennis Ritchie
- [https://www.youtube.com/watch?v=D3bblng-Kcc](https://www.youtube.com/watch?v=D3bblng-Kcc)
- [https://code.google.com/p/serialchart/](https://code.google.com/p/serialchart/)

Appendix

Program: The code running on arduino UNO
/*
Length of pendulum = 40 cm
Mass of pendulum rod = 11 grams,
Mass of weight = 6 grams
*/

#include <PID_v1.h>

int enc1 = 2; //Linear encoder pin 1
int enc2 = 4; //Linear encoder pin 2
int in1 = 3; //PWM output to L298n
int in2 = 11; //PWM output to L298n
int en = 12; //Pin to enable/disable H bridge
int count; //Counter for linear encoder (Resolution: 300ppi)
int abit; //Variable for reading angle sensor

double a_in, a_out, a_setp, p_in, p_out, p_setp, output, pwmin; //PID parameters
PID posPID(&p_in, &p_out, &p_setp, 0.001, 0.0, 0.0008, DIRECT); // PID loop for controlling the setpoint of pendulum
PID angPID(&a_in, &a_out, &a_setp, 4.6, 430, 0, DIRECT); // PID loop for controlling the motor

void setup()
{
    //TCCR2B = TCCR2B & 0b11111000 | 0x04;
    pinMode(enc1, INPUT);
    pinMode(enc2, INPUT);
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
    pinMode(en, OUTPUT);
    attachInterrupt (0, encoder, RISING); //Reading linear encoder for tracking cart position
    digitalWrite(en, HIGH); //Enable/Disable H-bridge
    Serial.begin(9600);
    posPID.SetMode(AUTOMATIC);
    posPID.SetSampleTime(10);
    posPID.SetOutputLimits(-3, 3);
    angPID.SetMode(AUTOMATIC);
    angPID.SetSampleTime(1);
    angPID.SetOutputLimits(-195, 195);
}

void loop()
{
    if(count < 1000 && count > -1000) //To prevent the cart from hitting the ends
    {
        p_setp = 0;
        p_in = count;
        posPID.Compute();
    
        a_setp = 545.4 + p_out;
        a_in = analogRead(abit);
        angPID.Compute();
        output = a_out;

        if(output<0)
        {
            pwmin = (-1*output) + 60;
            analogWrite(in2, 0);
            analogWrite(in1, pwmin);
        }
        if(output>0)
        {
            pwmin = (output+60);
            analogWrite(in1,0);
            analogWrite(in2, pwmin);
        }
        Serial.println(a_in);
    }
    else
    {
        digitalWrite(en,LOW);
        analogWrite(in1, 0);
        analogWrite(in2, 0);
    }
}

void encoder()
{
    if (digitalRead(enc1) == digitalRead(enc2))
    {
        count++;
    }
    else
    {
        count--;
    }
}

PCB layouts:

AS5043 Rotary Magnetic Sensor breakout board: KiCAD files can be found in the attachments with parts list

Front:
L298 Motor Driver PCB: KiCAD files can be found in the attachments with parts list

Front:

Back: