A modular inverted pendulum system for teaching and research

Ian S. Howard

Centre for Robotics and Neural Systems, University of Plymouth, Plymouth, UK.

Email: ian.howard@plymouth.ac.uk

Abstract

Here we describe a simple and elegant design for a modular inverted pendulum system, intended for teaching and as a research tool in the field of control engineering. This manual is to give you experience of operating the real inverted pendulum and using an Arduino Mega microcontroller to stabilize it.

1. Overview of the inverted pendulum

1.1. Introduction

An inverted pendulum is a system that consists of a rod that acts as the pendulum, which is pivoted at one end and attached to a cart (Fig. 1). The cart can travel backwards and forwards on some kind of linear track and the idea is that by moving the cart appropriately it is possible to balance the pole and keep it in its inverted position.

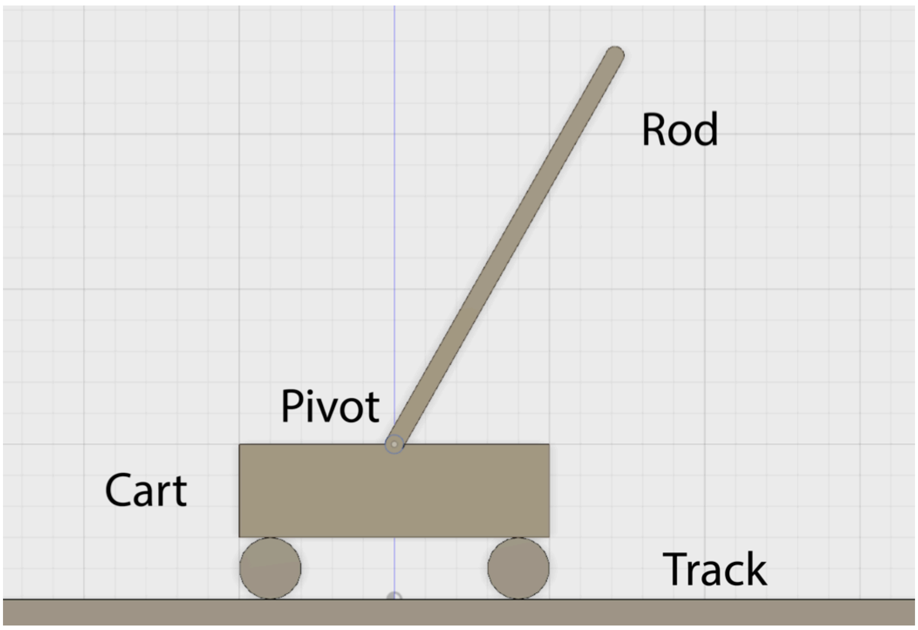


Figure 1. Schematic of a typical inverted pendulum. By moving the cart to the left and the right it is possible to balance the pole and keep it upright.
The pendulum has a stable position when it's hanging down normally. If slightly perturbed from this stable position it will swing back and forwards – i.e. oscillate. However, it also has an unstable position when it is standing upwards on the pivot. Balancing a pendulum in this upright position is often use as a classical example of controlled engineering and task is to make the system, which is inherently unstable, remain standing-up in the presence of slight disturbances such as light tapping. Once again this can be achieved by using appropriate control that measures the angle and moves the cart to stabilize the pole.

In typical implementations, the angle of the pendulum is measure at the joint us-ing either a potentiometer or angular encoder. The cart is also driven backwards and forwards along its track by means of some kind of electric motor.

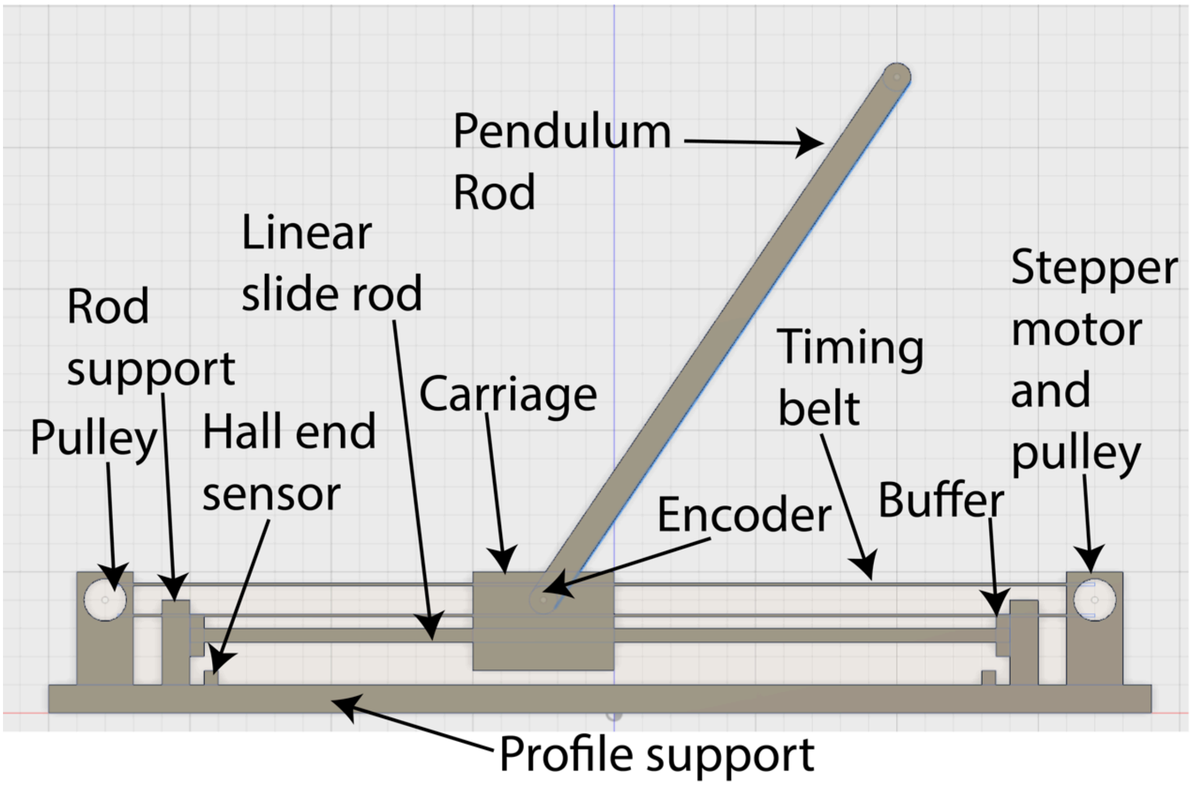


Figure 2. Schematic of the modular inverted pendulum design, illustrating its main components.

1.2. Equilibrium positions

An inverted pendulum system exhibits two equilibrium points. In its stable resting state the pendulum points downwards and the system behaves like a normal pendulum. E.g. like the pendulum in a pendulum clock. In this stable equilibrium point when it is perturbed by applying a small push, it will swing back and forth at a characteristic frequency determined by its dynamical properties. Damping in the joints and air resistance will ensure that the amplitude of this osculation reduces over time and after some time the pendulum will once again will up stationary at its equilibrium position.

In its inverted configuration, the pendulum is unstable without suitable control. The inverted pendulum is often used as a system to demonstrate how control systems can be used to stabilize systems that are inherently unstable.

1.3. Inverted pendulum system components

Here we adopt a modular approach to design. The system comprises several parts, which are illustrated in Fig. 2. The assembly and actuation are mounted on 20mm aluminum profile and attached by means of bolts and T-nuts. This construction technique greatly facilitates adjustment of the apparatus and also leads to a professional and aesthetically pleasing appearance. The frame of the system consists of two 20mm aluminum profile sections.

To provide a low friction tack, 16mm diameter stainless steel rods are attached to the profile frame on supports by means of aluminum clamps and mounting blocks. The large diameter rods were needed to ensure minimal bending when placed used the load of the carriage whilst swinging the rod around. Stainless steel was chosen to avoid problems with corrosion and entire the rods maintain a shiny appearance.

At the end of each rod section a protective rubber grommet was used to damp any collisions that may occur with the carriage and the end support blocks.

Hall effect sensors were located on the frame so that they would deactivate the motor drive when the carriage can close to the end of its travel in both directions. A neodymium magnet mounted underneath the carriage provided the necessary switching signal to achieve this.

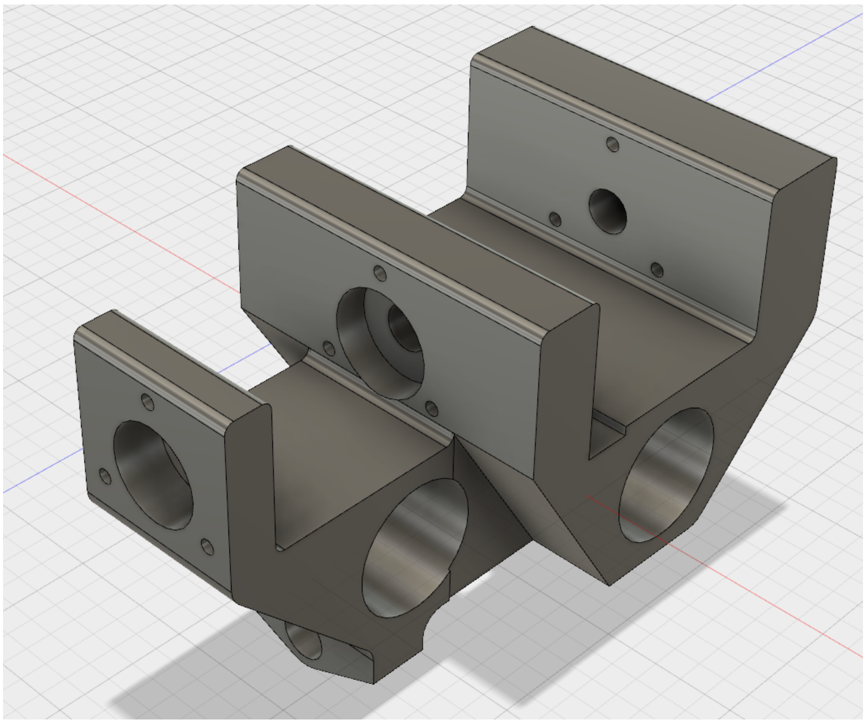


Figure 3. Pendulum carriage. The part is 3D printed from PLA. It accepts linear bearing at the base and ball bearing races to support the pendulum shaft at the top. It also has a recess to mount an incremental encoder. Underneath a cylindrical hole provides the mounting location for a neodymium magnet that is used to trigger the end-stop Hall sensors.

A carriage runs along the two rods and low friction is achieved by using linear ball bearings. The basic design of the carriage is shown in Fig. 3. It is moved along its rails by means of a GT2 timing belt.

A motor drive unit, consisting of a mount that fits in between the profile rails and supports a NEMA23 stepper motor attached by means of an angle bracket. The motor is currently driven from a Kuman 3D Printer Controller kit for Arduino RAMPS 1.4 Controller Board. It was operated from an Arduino Mega 2560 R3 Microcontroller programmed in C++.

A tooth GT2 pulley is attached to the motor to provide linear drive to the carriage. At the opposing end of the frame a mounting block holds a pulley so that the GT2 belt can run the length of the track and therefore be used to drive the carriage. Ball bearing were used in the pulley to ensure all moving parts could rotate with high precision and with little frictional resistance despite high belt tension.

The main track holders, motor drive and end pulley support are all separately attached to the aluminum profile structure with T-nuts end so they can easily be removed and exchanged with other similar components. This also provides a simple method for tensioning the belt by sliding the motor assembly backwards or forwards along the profile until the designed tension has been achieved. Fig. 4 shown the carriage and motor drive assemble close up.

The rod section is attached to a shaft mounted in the carriage. An important design requirement was to ensure it swings freely through 360 degrees and clears the profile frame structure. The shaft is supported by 2 sets of ball bearing and its far end is coupled to an incremental encoder. Fig. 5 shows the pendulum rod in its unstable inverted orientation.

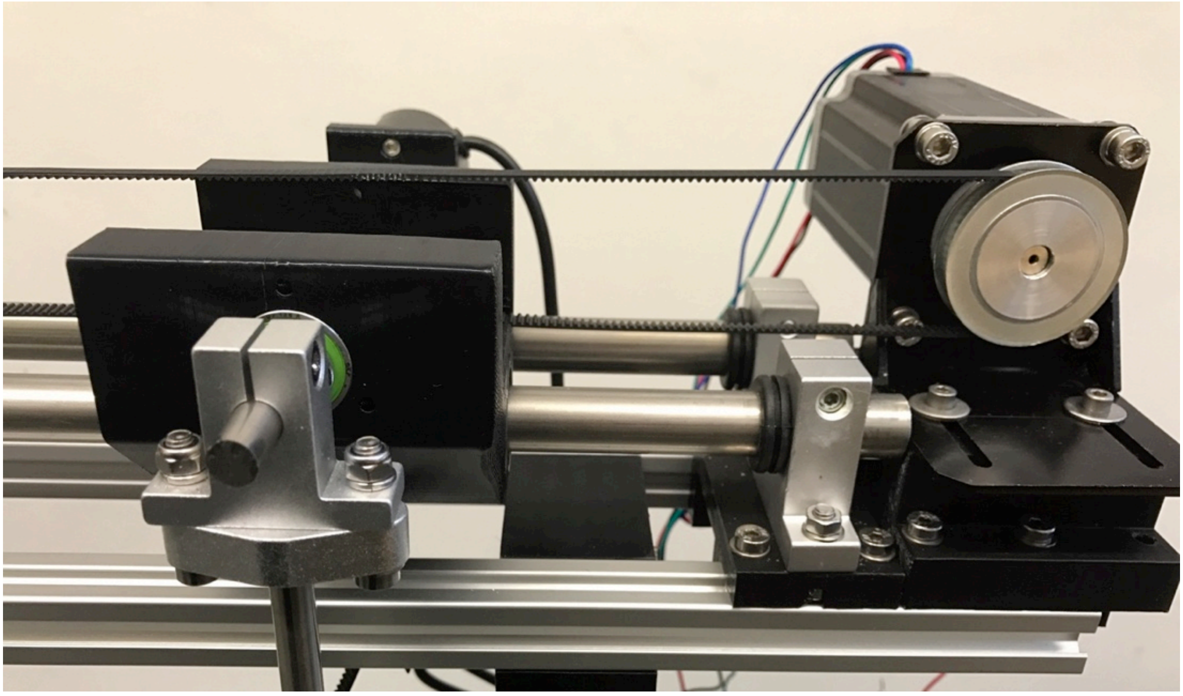


Figure 4. Pendulum drive motor and carriage mechanism. The drive pulley attached to the stepper motor can be seen on the right and the attachment clamp for the pendulum rod can be seen on the left.

1.4. Modular design supports task characteristics changes

The modular construction ensures that the parameters of the inverted pendulum system are easy to change. This is particularly useful in teaching scenarios because different tasks can then be given to different groups of students with only minor modifications to the apparatus. For example, the pole can be changed with one on a differ-ent length. The drive pulley can be exchanges with one with a different diameter div-ing different torque and speed transmission characteristic of the actuation mechanism.

Also, due to the modular design, the motor actuation block could be replaced with a DC motor thereby enabling the use of force control instead of position control. In addition, a potentiometer could be used instead of the incremental encoder, requiring the use of analogize position measurement rather than pulse counting.

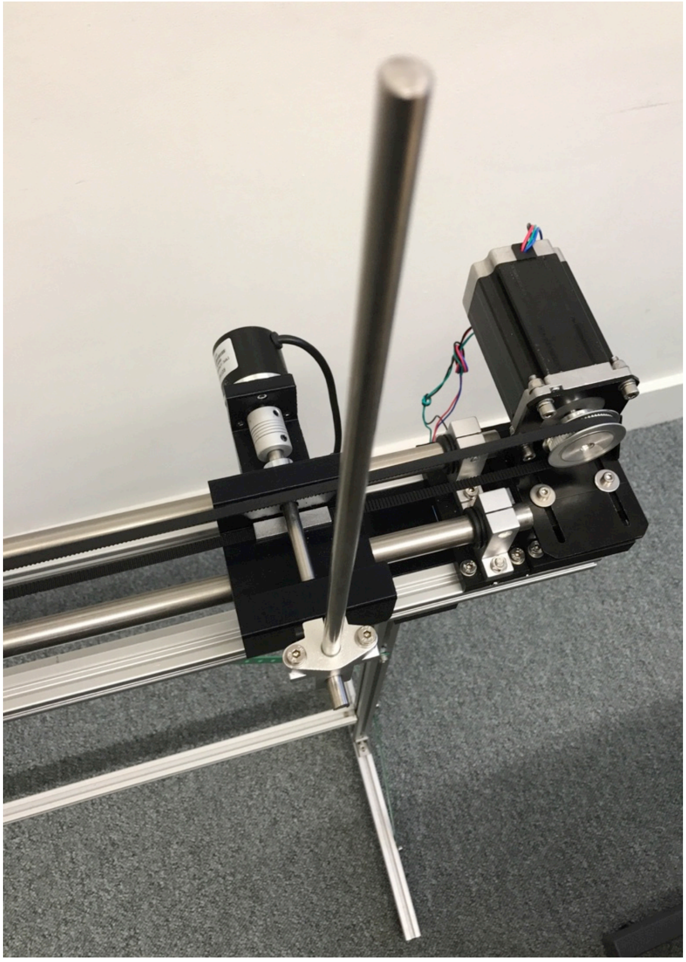


Figure 5. Pendulum system with the rod shown vertically in its unstable position

1.5. Construction techniques

AutoCAD Fusion 360 was used to design all the mounting blocks and the car-riage. This was used to generate STL format files and the mechanical parts were sub-sequently manufactured in PLA using a Flashforge Creator Pro 3D printer. We note that higher impact materials such as carbon-reinforced nylon would greatly improve the robustness of the design but in this initial prototype, PLA was considered an adequate choice.

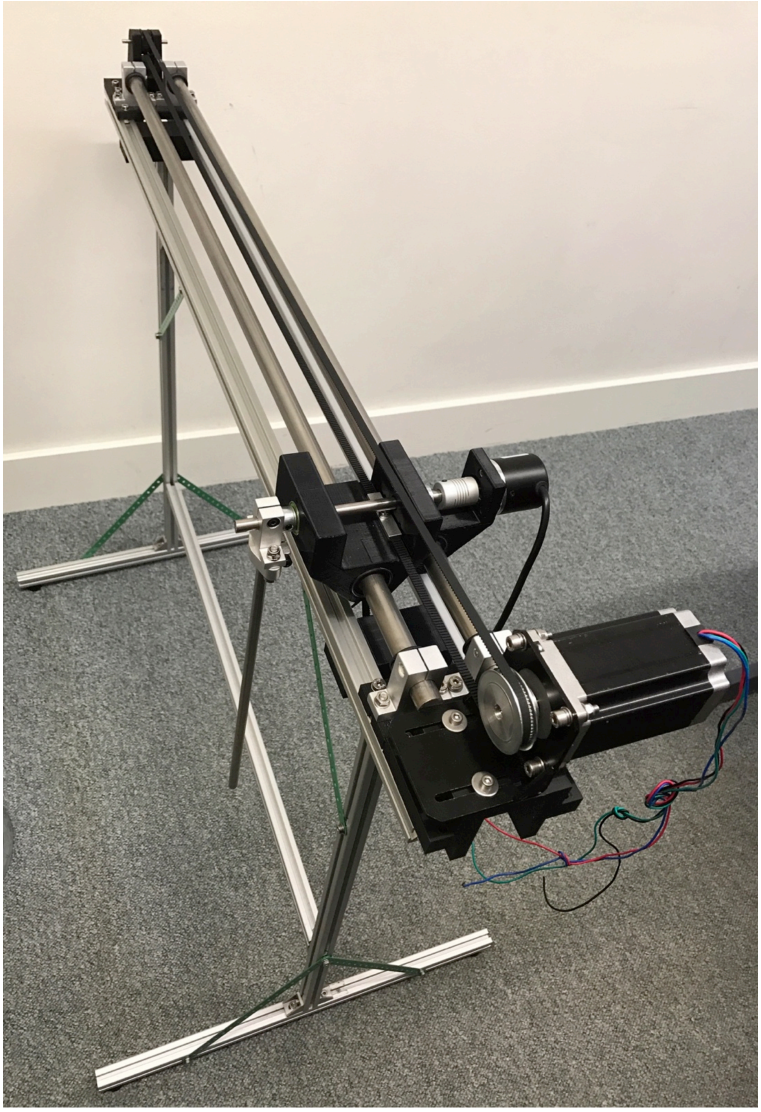


Figure 6. Pendulum system mounted on its aluminium profile stand. This provides an elegant means of support and allows the pendulum to rotate freely and avoid collisions.

1.6. The pendulum stand

Often inverted pendulums require some means of attachment to a table. Here it was felt a more elegant solution would be achieved by building a custom-made support. The stand to mount the pendulum system was also build out of 20mm aluminium profile, thereby reducing weight and making it was to transport (an important requirement for lecturers). This provides an elegant way to support it and consists of two pillars that fit in between the aluminium frame of the pendulum structure. The assembled pendulum on its stand is shown in Fig. 6.

2. State feedback control of inverted pendulum

2.1. State space analysis of pendulum dynamics

The linearized differential equation describing the inverted pendulum kinematics is given by

$$\left(I+ml^2\right)\frac{d^2\theta}{dt^2}+\mu\frac{d\theta}{dt}=mgl\theta+ml\frac{d^2x_p}{dt^2}$$

Where:
The angle to the vertical is denoted by θ
The coefficient of viciously is denoted by μ
The mass of the pendulum is denoted by m
The moment of inertial of the rod about the center of mass is denoted by I
The length to the centre of mass is denoted by *l*
The displacement of the pivot is given by x_p
We note that this kinematic description is sufficient to derive control provided we only use cart velocity as the control input. If we instead want to control the cart using applied force (which is often done in many inverted pendulum implementations) then we would also need to include and make use of an additional force equation.
Re-writing the differential equation describing the inverted pendulum

$$\Rightarrow\frac{d^2\theta}{dt^2}=\frac{-\mu}{\left(I+ml^2\right)}\frac{d\theta}{dt}+\frac{mgl}{\left(I+ml^2\right)}\theta+\frac{ml}{\left(I+ml^2\right)}\frac{d^2x_p}{dt^2}$$

So, we can stabilize the pendulum using velocity control, we write

$$\frac{d^2x_p}{dt^2}=\frac{dv_c}{dt}$$

This leads to the equation

$$\Rightarrow\frac{d^2\theta}{dt^2}=\frac{-\mu}{\left(I+ml^2\right)}\frac{d\theta}{dt}+\frac{mgl}{\left(I+ml^2\right)}\theta+\frac{ml}{\left(I+ml^2\right)}\frac{dv_c}{dt}$$

Let the constant terms be represented by the coefficients

$$a_1 = \frac{\mu}{\left(I + ml^2\right)}$$

$$a_2 = \frac{-mgl}{\left(I + ml^2\right)}$$

$$b_0 = \frac{ml}{\left(I + ml^2\right)}$$

$$\Rightarrow \frac{d^2\theta}{dt^2} = -a_1 \frac{d\theta}{dt} - a_2\theta + b_0 \frac{dv_c}{dt}$$

Choosing state space representations

$$x_1 = \theta$$

$$\Rightarrow \dot{x}_1 = \frac{d\theta}{dt}$$

$$x_2 = \frac{d\theta}{dt} - b_0 v_c$$

$$\Rightarrow \frac{d\theta}{dt} = x_2 + b_0 v_c$$

$$\Rightarrow \dot{x}_1 = x_2 + b_0 v_c$$

$$\Rightarrow \dot{x}_2 = \frac{d^2\theta}{dt^2} - b_0 \frac{dv_c}{dt}$$

From before

$$\frac{d^2\theta}{dt^2} = -a_1 \frac{d\theta}{dt} - a_2\theta + b_0 \frac{dv_c}{dt}$$

$$\Rightarrow \frac{d^2\theta}{dt^2} = -a_1\left(x_2 + b_0 v_c\right) - a_2 x_1 + b_0 \frac{dv_c}{dt}$$

Substituting into

$$\Rightarrow \dot{x}_2 = \frac{d^2\theta}{dt^2} - b_0 \frac{dv_c}{dt}$$

$$\Rightarrow \dot{x}_2 = -a_1\left(x_2 + b_0 v_c\right) - a_2 x_1 + b_0 \frac{dv_c}{dt} - b_0 \frac{dv_c}{dt}$$

$$\Rightarrow \dot{x}_2 = -a_1 x_2 - a_2 x_1 - a_1 b_0 v_c$$

From the state space representations

$$\dot{x}_1 = x_2 + b_0 v_c$$

$\dot{x}_2 = -a_1 x_2 - a_2 x_1 - a_1 b_0 v_c$

State space notation takes the form

$$\dot{X} = AX + BU$$
$$Y = CX + DU$$

Writing in matrix format we therefore have

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -a_2 & -a_1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_0 \\ -a_1 b_0 \end{bmatrix} v_c$$

$$\Rightarrow \frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{mgl}{\left(I+ml^2\right)} & -\frac{\mu}{\left(I+ml^2\right)} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \frac{ml}{\left(I+ml^2\right)} \\ \frac{-\mu ml}{\left(I+ml^2\right)^2} \end{bmatrix} v_c$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Where output y is the pendulum angle θ. The state space system matrices as therefore

$$A = \begin{bmatrix} 0 & 1 \\ \frac{mgl}{\left(I+ml^2\right)} & -\frac{\mu}{\left(I+ml^2\right)} \end{bmatrix}$$
$$B = \begin{bmatrix} \frac{ml}{\left(I+ml^2\right)} \\ \frac{-\mu ml}{\left(I+ml^2\right)^2} \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

2.2. Augmenting positional state

In practice, we want to control cart position as well as angle and angular velocity since otherwise it might never stop moving. If we do so we can control the cart position to remain at zero (its initial location). To achieve this, we add a third state x_3

to represent cart position. Since the control signal is cart velocity, the differential of x_3 is simply given by the input velocity control signal. Therefore, we can write:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -a_2 & -a_1 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_0 \\ -a_1 b_0 \\ 1 \end{bmatrix} v_c$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Substituting in value for the coefficients leads to the expression

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{mgl}{\left(I+ml^2\right)} & -\frac{\mu}{\left(I+ml^2\right)} & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} \frac{ml}{\left(I+ml^2\right)} \\ \frac{-\mu ml}{\left(I+ml^2\right)^2} \\ 1 \end{bmatrix} v_c$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Therefore, we can write the augmented state space system matrices as

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{mgl}{\left(I+ml^2\right)} & -\frac{\mu}{\left(I+ml^2\right)} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{ml}{\left(I+ml^2\right)} \\ \frac{-\mu ml}{\left(I+ml^2\right)^2} \\ 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

It is easy tocompute the numeric values of the matrices using Matlab.

2.3. State feedback velocity control

To implement feedback control of this system we must use some form of feedback in the system. One option is to use state feedback. This is represented by the expression
$U = -KX$

The feedback matrix K represents feedback gain of the system state and R(t) represent a reference input

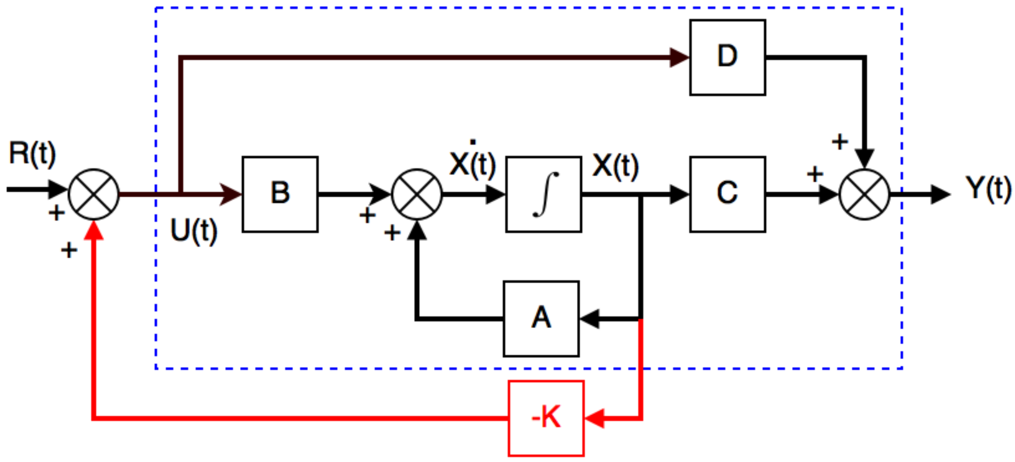


Figure 7. Signal flow diagram for a linear dynamical system operating under direct state feedback control. Dotted line shows the state feedback path, which includes multiplication by the feedback gain -K.

Substituting in feedback term to state space equations gives

$$\Rightarrow \dot{X} = AX + BU = AX - BKX = (A - BK)X$$
$$\Rightarrow Y = CX + DU = CX - DKX = (C - DK)X$$

Looking at the expression

$$\dot{X} = (A - BK)X$$

We see than the close- loop stability now depends on eigenvalues λ given by the characteristic equation

$$\left| I\, I - \left(A - BK \right) \right| = 0$$

Thus, we can influence the location of the eigenvalues of the system by changing the gain matrix K

2.4. Luenberger observer

We now need to choose Luenberger estimation gain L

$$\dot{\hat{X}} = A\hat{X} + BU + L(y - C\hat{X})$$

We can choose observer the gains L using computation, e.g. Matlab. This can be done by pole placement or using optimality criteria. Pole placement can be achieved using the Ackermann formula. NB: Here we only use an observer to estimate theta and thetaDot since we can directly estimate position by integrating the velocity control signal.

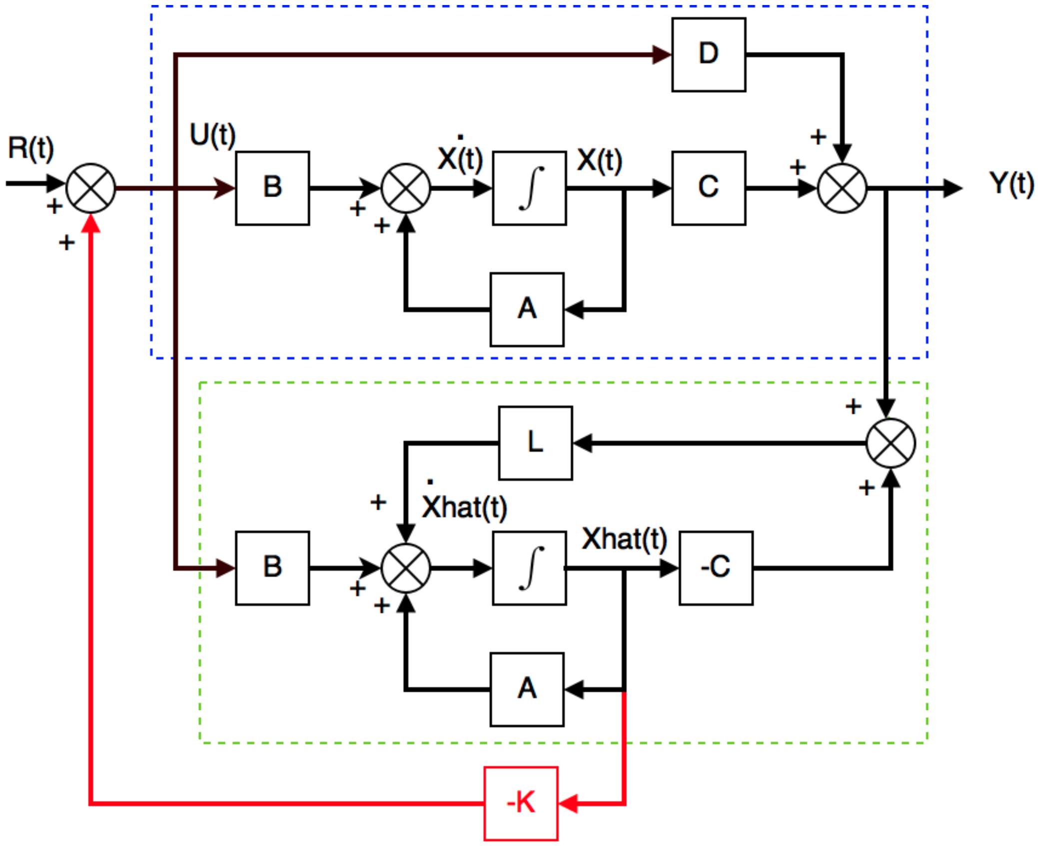


Figure 8. Using a feed-forward observer to estimate plant state. The observer is simply a model of the plant dynamics and generates a state estimate Xhat, which can be used for state feedback instead of using the actual system state X. Feed forward state estimation required an accurate model of the plant and this approach is not generally used.

3. Inverted pendulum controller

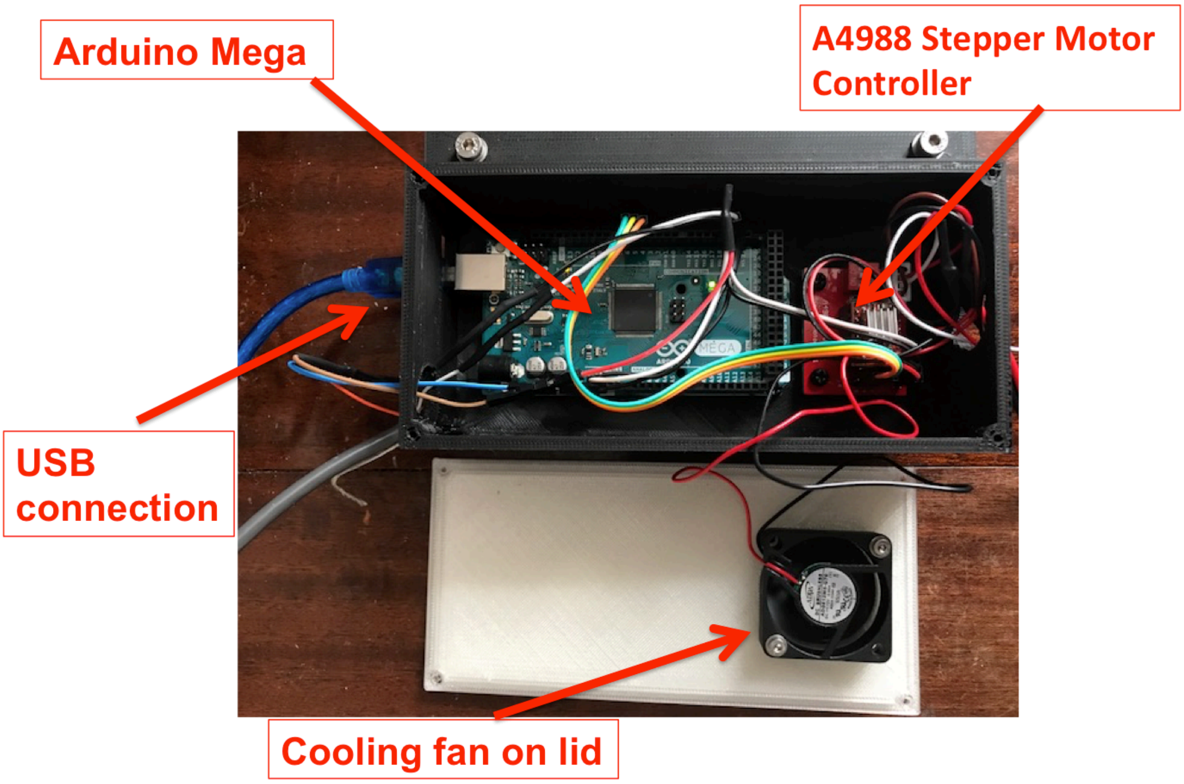


Figure 9. Arduino Mega based stepper motor controller unit.

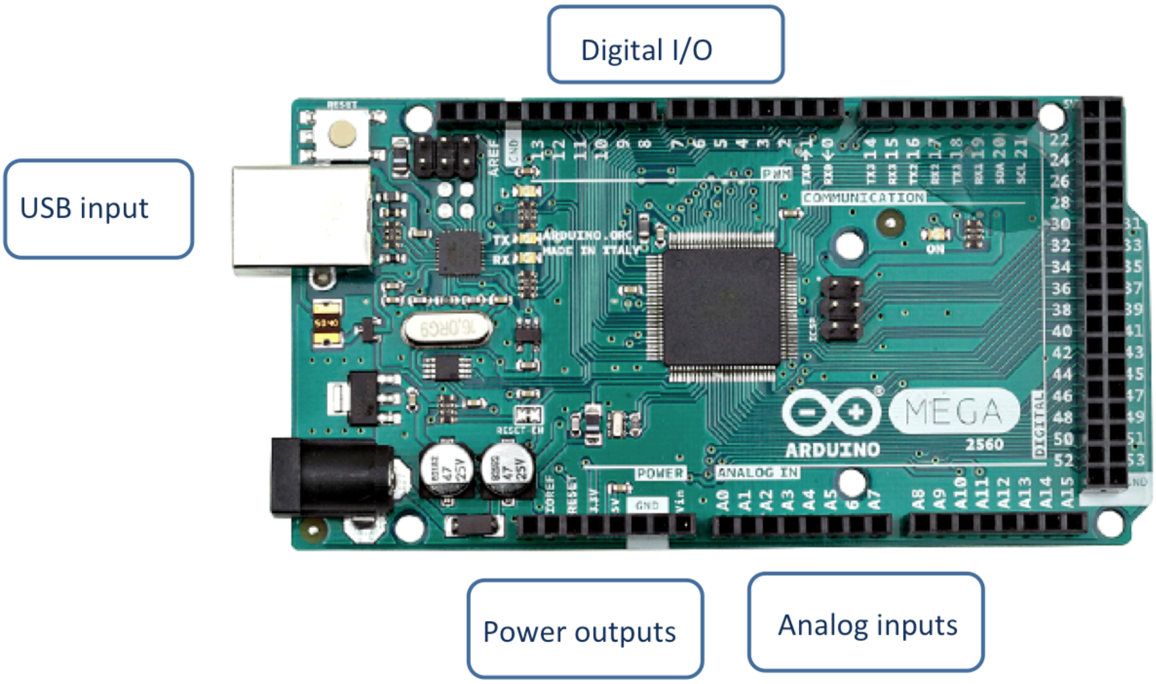


Figure 10. Arduino Mega connections

4. Operating the inverted pendulum

4.1. Connect up the Arduino pendulum unit

Connect the host PC to the Arduino unit by means of a USB cable.

Connect up the power lead to the special pendulum bench power supply set to 18V and 2A.

Do not use the standard electronic workshop powersupply as they can generate voltages that could destroy the unit.

Connect the red wire to the +ve terminal and the black wire to the 0v terminal.

Take care not to connect this leads the wrong way around since this will destroy the controller!

Please do not use more than a supply of 18V as this will also destroy the controller!

4.2. Example programs

A suite of example programs is available the zip file ROCO218Arduino.zip

Expand the file. It contains 5 directories containing Arduino applications and also a libraries directory. Copy all folders in the libraries directory into the default Arduino libraries directory on you PC.

All the Arduino application programs display a help menu if when the character ‘h’ is entered on the serial monitor. These application programs are in the following directories.

ProcessControlTest: This implements a menu driven command manager running over the serial monitor

SimpleHallSensorTest: This implements end stop detection using the Hall sensors

SimpleEncoderTest: This reads to encoder value and print out via the serial output. If you select the plot monitor you can generate a plot of the signal versus time.

StepperVelocityTest: This uses a velocity control stepper motor class to drive the carriage back and forth sinisoidally.

SFCInvPendTX: This implements SFC control of an inverted pendulum with the state vector augmented with cart position.

4.3. Arduino program structure

There are three basic stages of operation of an Arduino program

1. Initialization and definitions
   This is where variables and functions used by the algorithm are defined and initialized
   Class objects are also build at this stage of operation.

2. Setup
   The setup phase is where flags are iinitialized and connecrs o the host PC are setup.

3. Loop

   The loop is an endless possing loop that is used to service the programs needs, Perioidc reading and writing from I/O can be carried out here,

In the following sections, the processing used to implement each of the suite of demonstation programs is explained in terms of these processing phases using flow charts.

4.4. Menu control demo program

It can be useful to have control of a microcontroller using a host program that communicated with it. In an example program, we use the Arduino serial monitor to control a command menu on the Arduno Mega. Upload the program ProcessControlTest.ino and familiarize yourself with its operation.

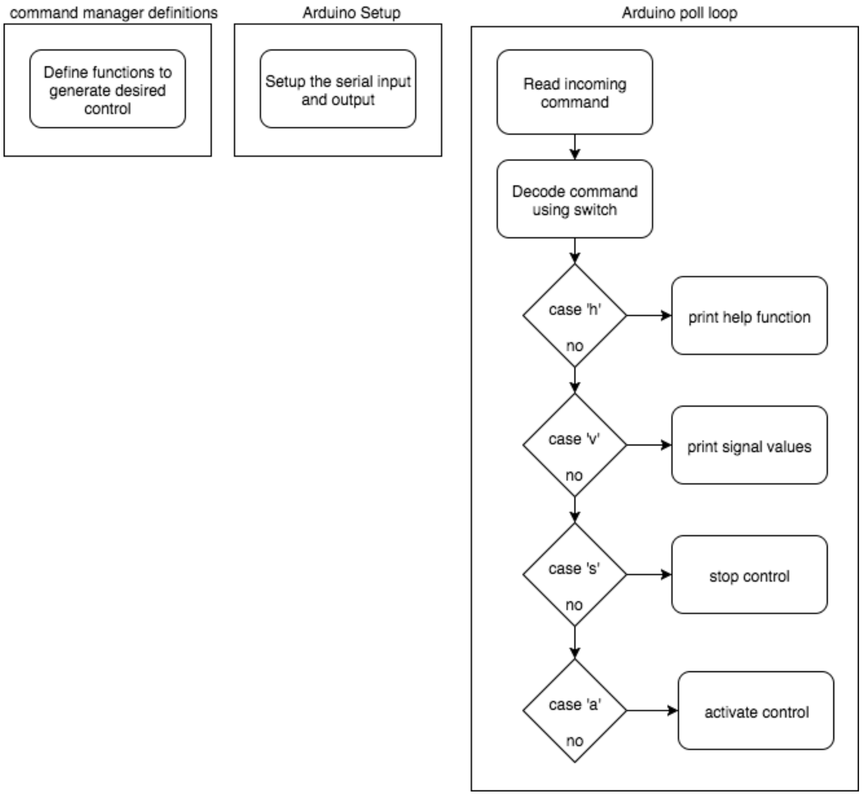
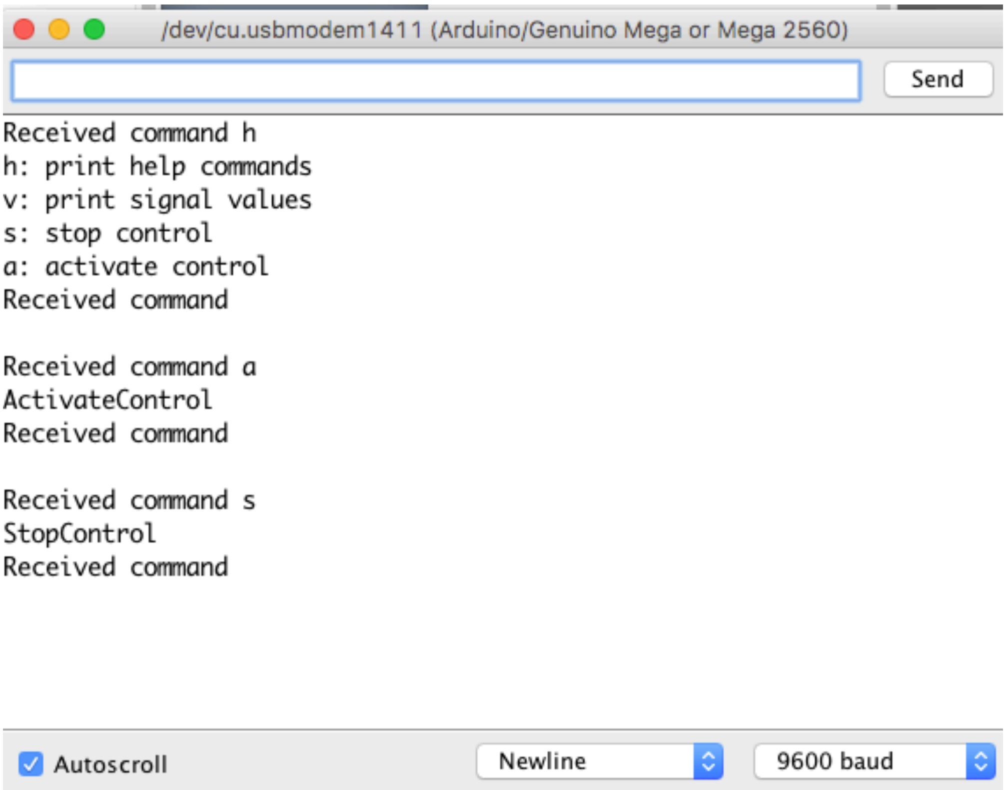
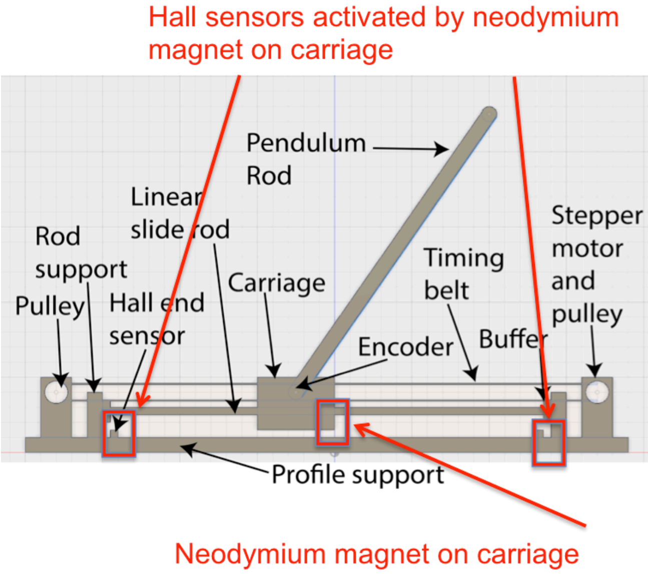


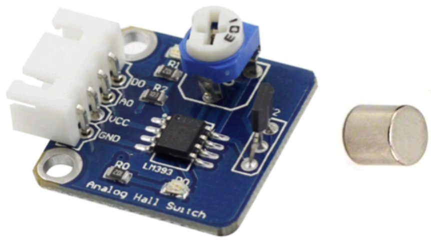
Figure 12. Menu demonstration program decision flow chart



4.5. Hall end stop sensors demo program

Many real-time robotic systems make use of end stop sensors to terminate operation if dangerous conditions arise. In the case of an inverted pendulum, we need to avoid the carriage hitting its end positions when it reaches its end of travel to the left and to the right. To achieve this, magnetically activated Hall sensors are used. This simple switch activated by the application of a magnetic field. Upload the program SimpleHallSensorTest.ino and familiarize yourself with its operation.





SunFounder Analog Hall Switch Sensor Module
0v to GND
+5v power to VCC
Digital output on D0

Figure 13. Hall sensor locations

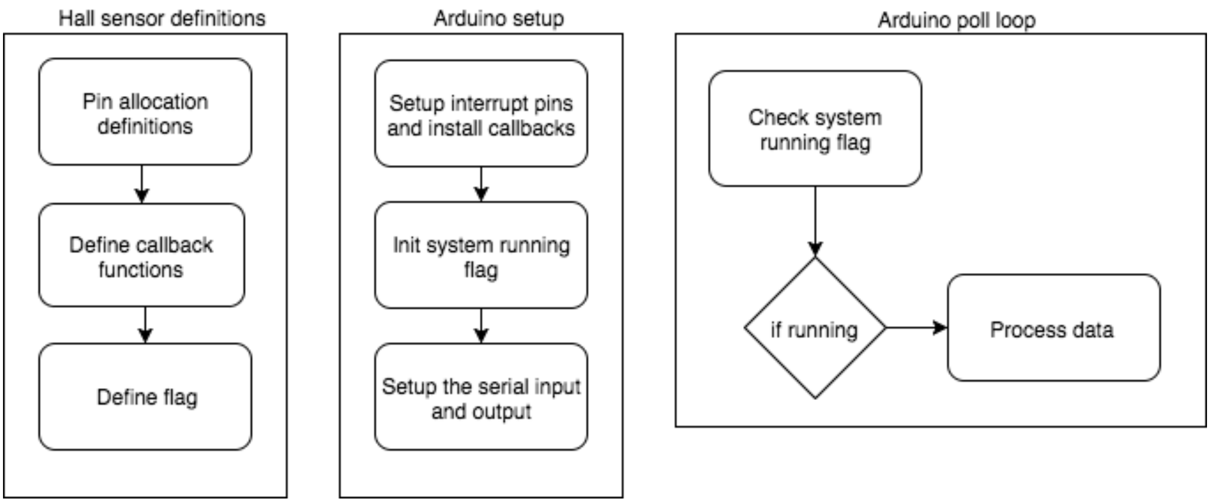


Figure 14. Hall sensor demonstration program decision flow chart

4.6. Incremental encoder demo program

One method to measure pendulum position is to use an incremental encoder. It generates a relative position signal suitable for positioning tasks.

Here we use a 600ppr or a 2000ppr encoder that runs in conjunction with a simple encoder class that make use of all pulse edges and therefore achieved 4x resolution. Upload the test program SimpleEncoderTest.ino and familiarize yourself with its operation. You can use the serial plotter to plot a graph of the encoder values against time and thereby visualise the amplitude decay of the pendulum when disturbed from its normal position.

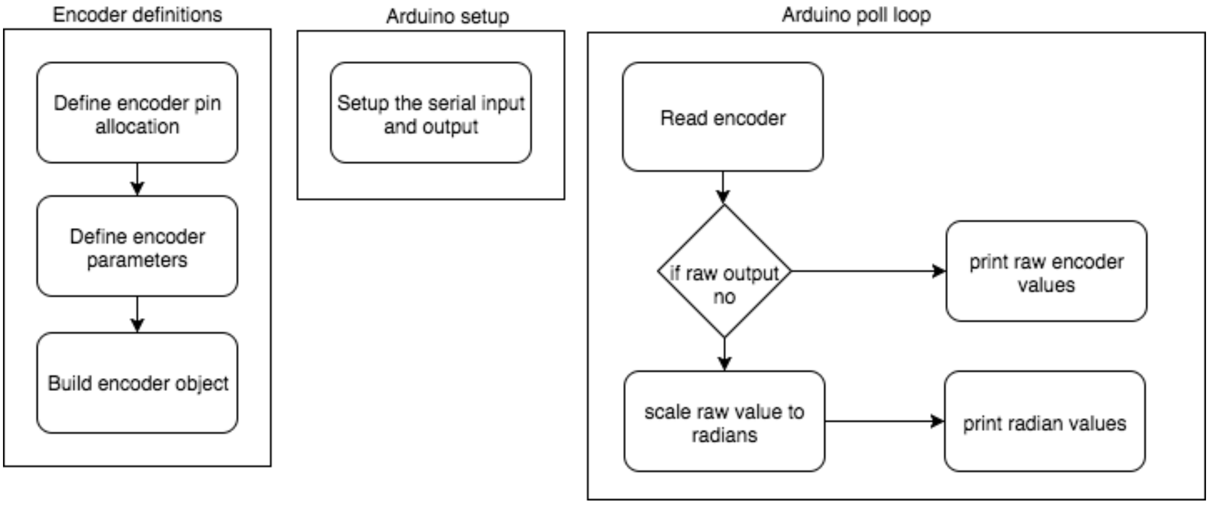


Figure 14. Incremental encoder demonstration program decision flow chart

Plot output using serial monitor window provided but eh Arduino integrated delevopment environment is shown in Fig, 17,

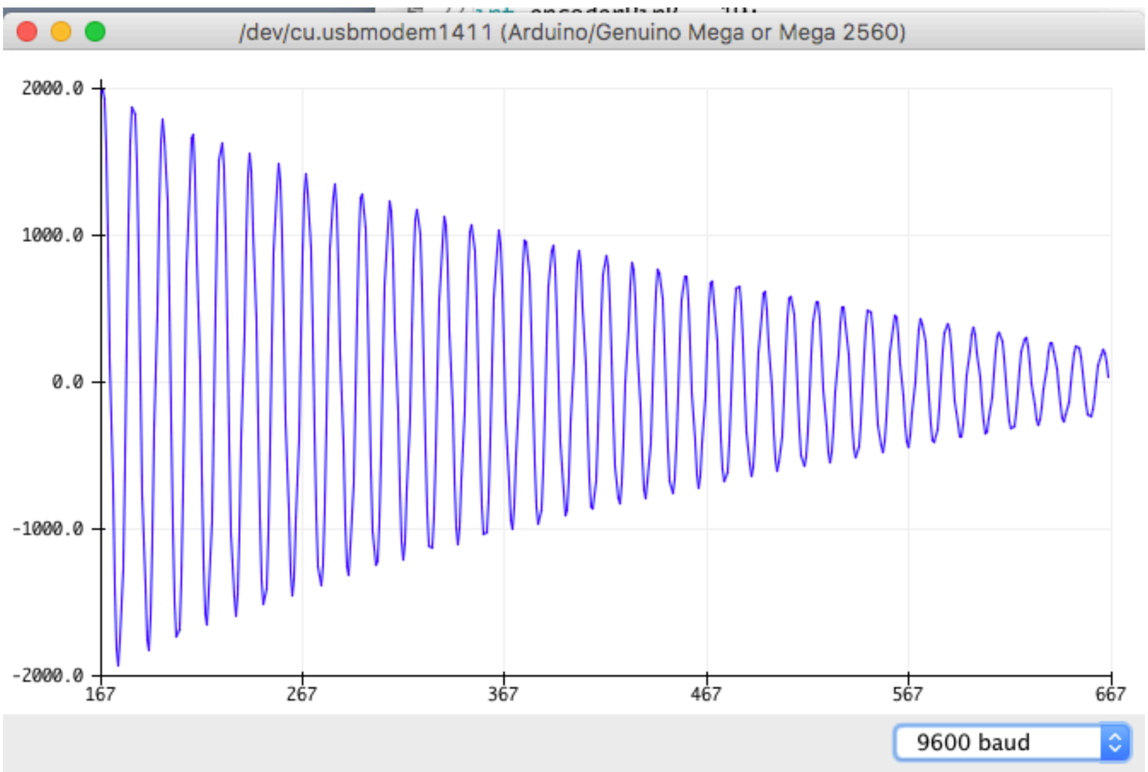
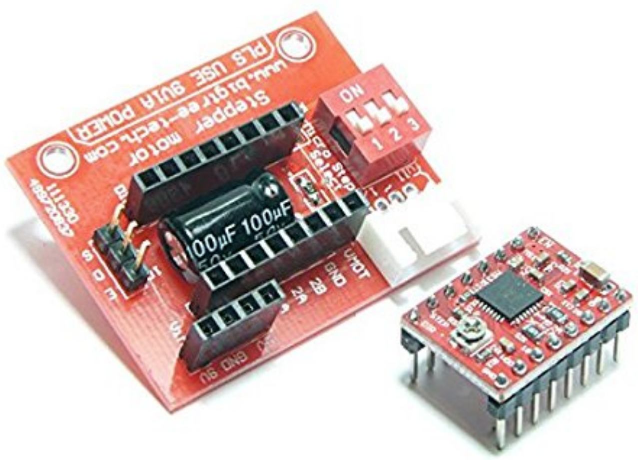


Figure 15. Incremental encoder output plotted ising the Arduino plot window.

4.7. Stepper motor velocity control demo program

One method to control an inverted pendulum is to control the velocity of the carriage using a stepper motor. Here we use micro step mode, which implements a gradual transition between pole positions. This leads to smoother operation that full-step operation.



The A4988 Stepper Motor Controller is used to drice the spepper motor. This implements intelligent current control up to 2A. We set stepping mode via control pins – and here is is set to 4x micro stepping. It operatesd using a simple step and direction control interface. To operate a stepper motor using this controller, we need to generate control pulses (1 pulse per micro step) direction signals (so motor turns in desired direction). For more information, we https://www.pololu.com/product/1182.

To exprience how a stepper motor can operate under velocity control, upload the program StepperVelocityTest.ino and familiarize yourself with its operation. This program will dive the carriage in a sinusoidal fashion. Try adjusting the program amplitude and frequency and observe the result.

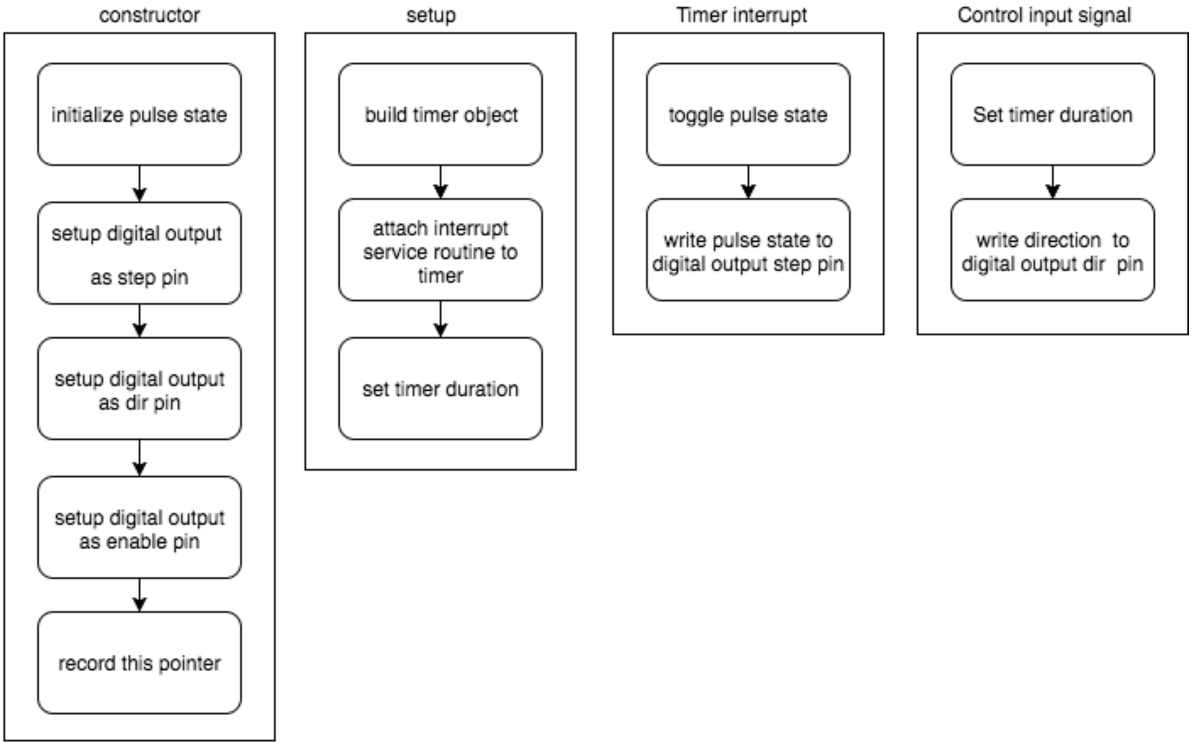


Figure 15. CStepper velocity control class flow chart

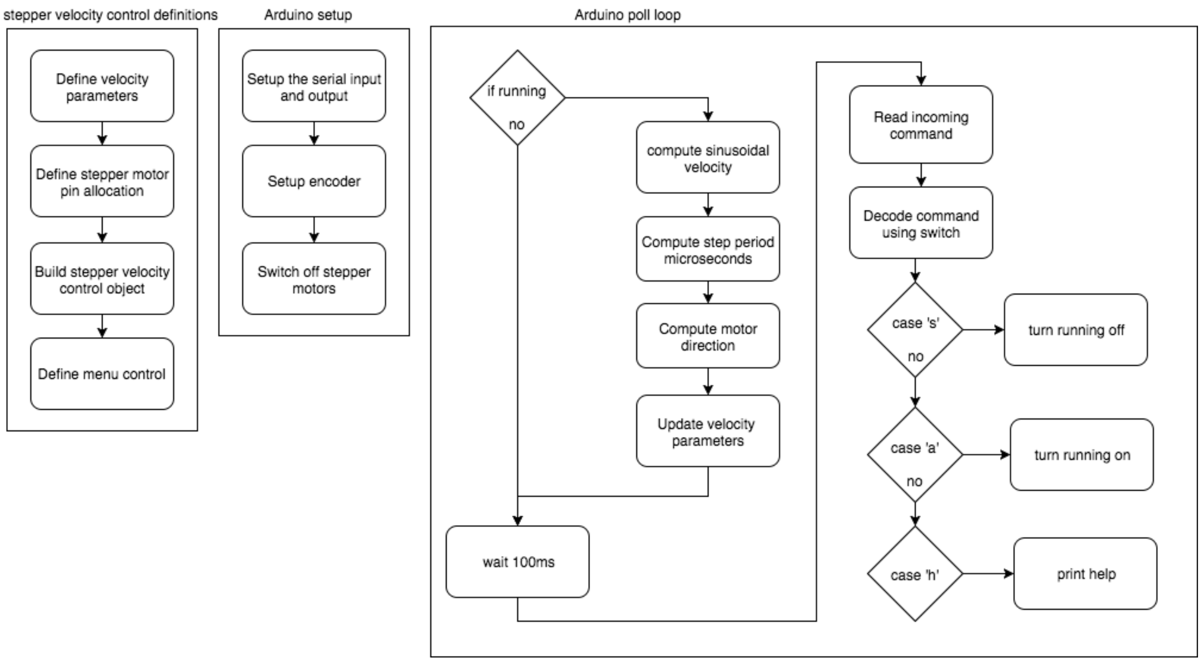


Figure 16. Arduino stepper velocity control flow chart flow chart

4.8. SFC of an inverted pendulum demo program

One method to control an inverted pendulum is to control the velocity of the carriage using a stepper motor. SFC velocity control of the inverted pendulum is implemented in the program SFCInvPendTX.ino. The control class is in the file CSFC3.cpp and tits operation is shown in Fig. 17. Run the program and investigate the effect of changing the state space matrices, observer and SFC gains. An overview of the operation of the main Arduino program is shown in the flow graph in Fig. 18.

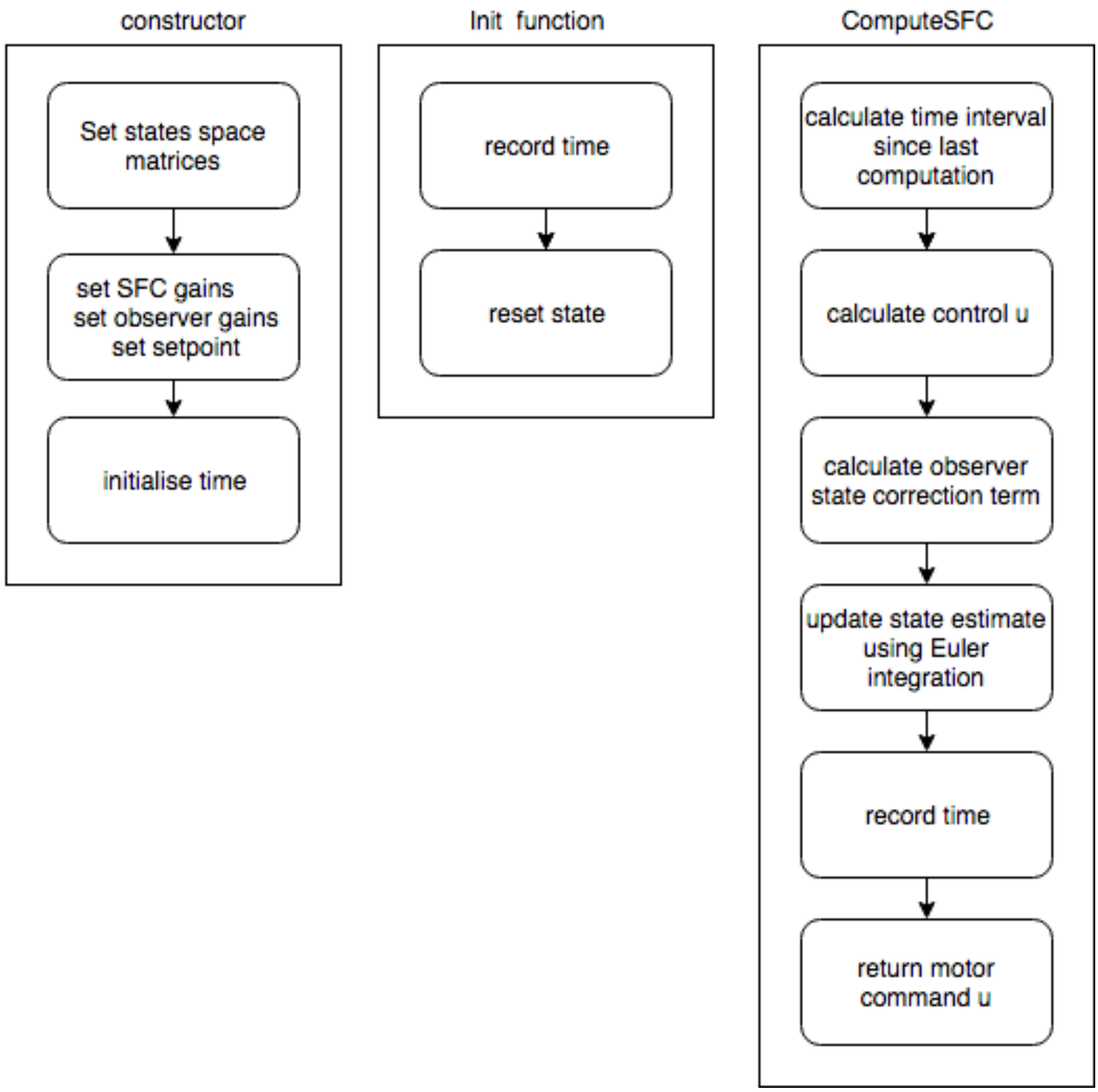


Figure 17. Arduino state feedback controller CSFC3 class flow chart

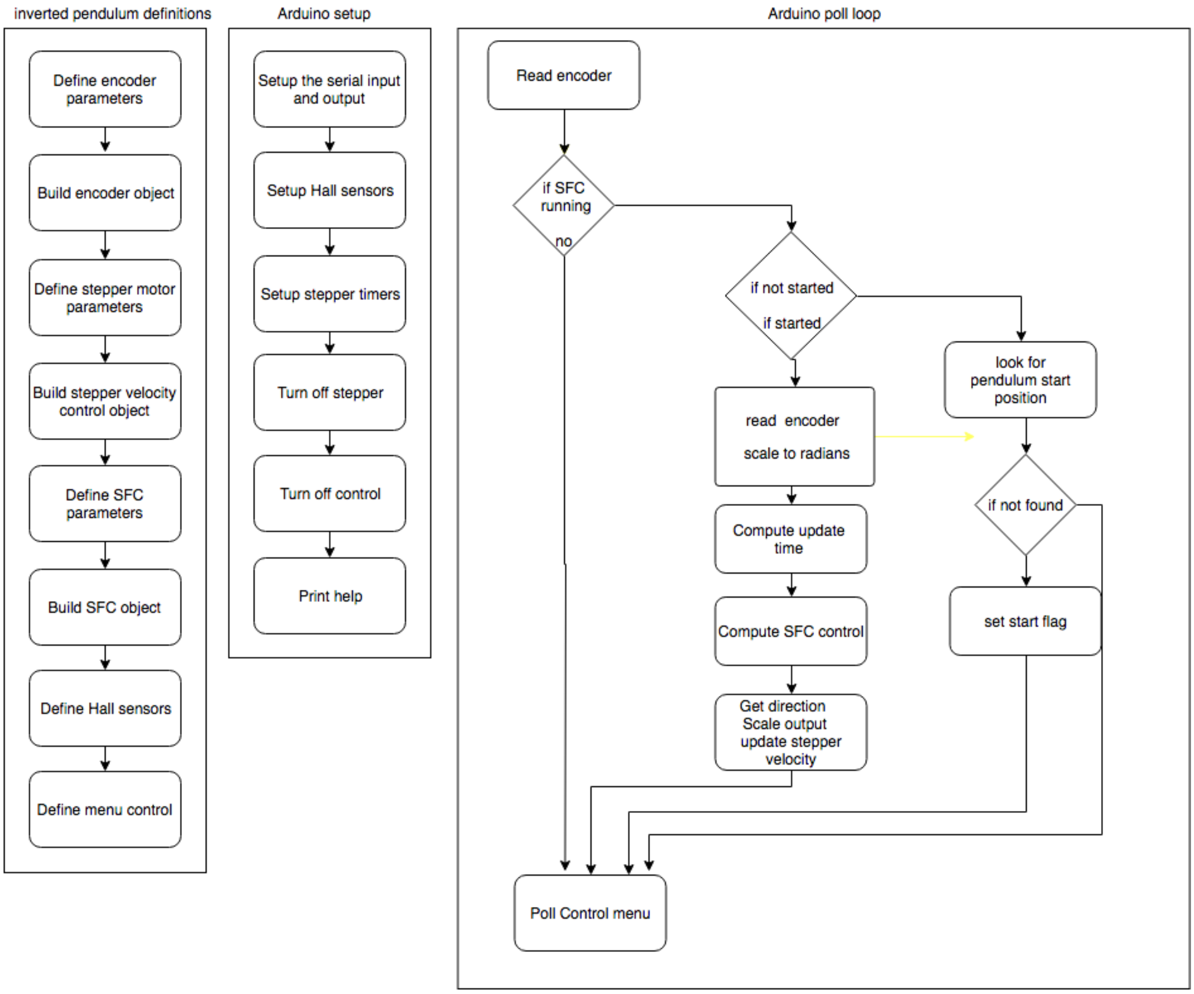


Figure 18. Arduino SFC inverted pendulum flow chart