UNIVERSITY OF HERTFORDSHIRE

Faculty of Engineering & Information Science

MASTER OF ENGINEERING

DEGREE WITH HONOURS

IN

EMBEDED INTELLIGENCE SYSTEM

Project Report

INVERTED PENDULUM
WITH ANFIS CONTROLLER

Ye Zhang

August 2005

DECLARATION STATEMENT

I certify that the work submitted is my own and that any material derived or quoted from the published on unpublished work of other person has been duly acknowledged. (ref. UPR AS/C/6.1, Appendix I, Section 2 – Section on cheating and plagiarism)

Student Full Name:

Student Registration Number:

Signed:

Date:

Abstract

This report describes how to generate and implement an ANFIS (Adaptive Neuro-Fuzzy Inference System) controller for an inverted pendulum system.

ANFIS, as a modern artificial intelligence system, combines the fuzzy logic and neural network to accomplish self-learning ability, which is more suitable to deal with the complicated non-linear systems. This objective has been achieved in Matlab simulation. The ANFIS controller, which is generated and trained by the data derived from another successful controller, is competent in keeping an inverted pendulum system in dynamic balance.

Also, this report focuses on the real hardware system design. All the components are selected and investigated. The whole hardware structure has already been built up and partially tested and improved. Especially, the Z8 microcontroller is studied and configured for this particular application.

Acknowledgements

Many thanks to my Project Supervisor Dr David Lee who offered valuable support and suggestions, as well as the encouragement at all stages throughout the project.

I would like to express my gratitude to Mr. John Wilmot, who helped a great deal during the practical work of this project.

Also many thanks to my friends and classmates for their support and assistance, in particular, Johnason, Sahil and Praveen.

As a memory of my one-year life in U.K.

List of Figures

Glossary

ACE	Absolute Contacting Encoder
ANFIS	Adaptive-Neural-Fuzzy Inference System
CPU	Central Processing Unit
DC Motor	Direct Current Motor
FIS	Fuzzy Inference System
GPIO	General Purpose Input/Output
GUI	Graphical User Interface
IC	Integrated Circuit
LED	Light-Emitting Diode
MCU	Micro-Controller unit
MF	Membership Function
PID	Proportional-Integral-Derivative
PWM	Pulse Width Modulation
RAM	Random Access Memory
UART	Universal Asynchronous Receiver/Transmitter

Contents

Chapter 1 INTRODUCTION

1.1 Background

1.1.1 Inverted Pendulum

Inverted pendulum, also called "pole on a cart", is a classic inherently unstable system (*Fig 1*). An inverted pendulum system consists of four inputs (pole angle θ, pole angle velocity $\dot{\theta}$, cart position x and cart speed $\dot{x}$) and single output (control force f). By applying a sequence of right and left forces to the cart, the pendulum can be balanced in upright and the cart works dynamically in the center of the rails.

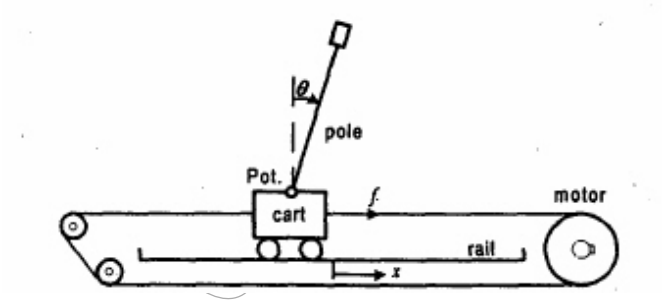


Fig 1.1 Inverted pendulum

1.1.2 Control Methods in Inverted Pendulum

Traditionally, the controller design is based on the simplified linear model of an inverted pendulum system, and the control force *f* is assumed to be a function of the four state variables. Classical control theories have been proven successfully in inverted pendulum control, such as PID (Proportional-Integral-Derivative) control method, but most of them require considerable knowledge of the accurate system dynamic model and complicated mathematic analysis.

Nowadays, with the development in Artificial Intelligence, it is not necessary to build up any confusing non-linear kinematic model. A hot research direction is using learning optimization algorithms like neural-network to achieve the self-adaptive ability. ANFIS (adaptive-neural-fuzzy inference system), as a hybrid intelligent system, can construct an input-output mapping based on both human knowledge (in the form of fuzzy if-then rules) and stipulated input-output data pairs.

1.2 Project Aims and Objectives

The overall aim of this project is developing a controller using ANFIS algorithm to balance an inverted pendulum system in dynamic stable.

To accomplish this aim, the following objectives have to be achieved:

- Learn the theory of ANFIS, as well as the fundamental of Fuzzy Logic and Neural Network. Understand the principles of self-learning. Know where and how to gather the training data and how to generate the ANFIS controller.
- Do simulation in Matlab®, and achieve a successful ANFIS controller with optimal performance.
- Do research on Z8 Encore!® Z8F642x Development Board, and study the specification of Z8 Encore!® MCU and know how to configure it for this particular application.
- Understand PWM together with H-Bridge techniques, which is a widely used method for motion control (DC motor control). Also, become familiar with the two sensors: Optical Encoder and Gray Code.
- Design the whole hardware system, and assemble all the components. After simulating the controller in Simulink®, test it in this real system.
- Learn C Language, and try to implement the ANFIS controller by writing the programs in ZDS II® (software with C compiler provided by ZiLOG®).

1.3 Overview of the System

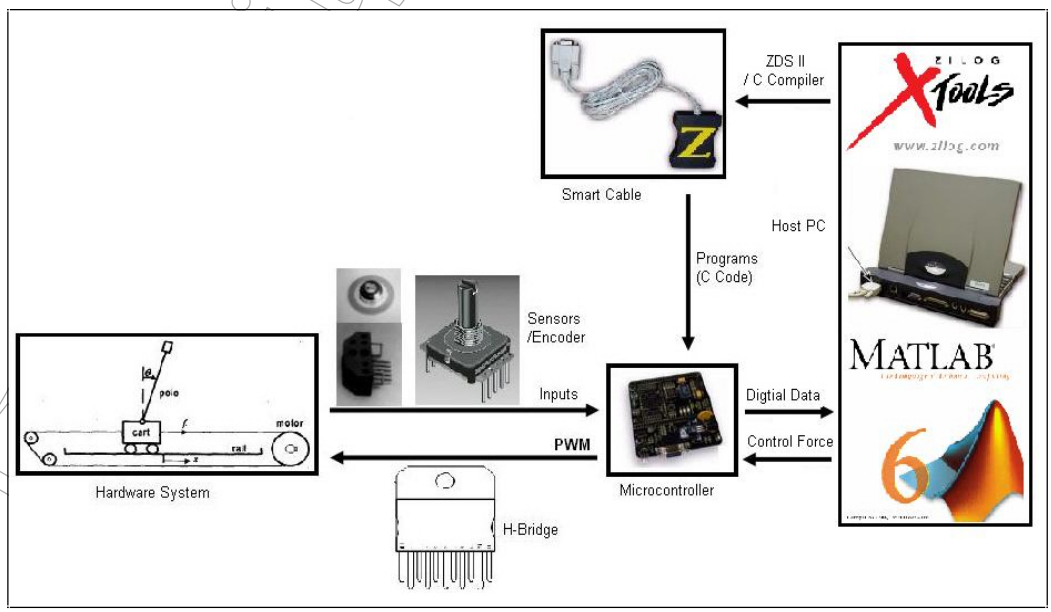


Fig 1.2 An overview of the system

The overall configuration: (*Fig 1.2*)

- It is a real-time controlling system.
- ANFIS controller is running in Matlab® in host PC.
- The hardware system is governed by Z8 Encore!® microcontroller, which is integrated onto the development board.
- All the essential programs in C code are debugged and compiled in the software ZiLOG Developer Studio II in host PC and transferred onto the Flash Program Memory on the development board, though the "Smart Cable".
- Z8 microcontroller interfaces the hardware system with the software controller. The state parameters of both pole and cart are read by the two sensors, and transmitted by Z8 into the ANFIS controller. Then the desirable output is generated by the controller and sent back to Z8 to be modulated into PWM signal.
- All the communication between Z8 MCU development board and the ANFIS controller is built up by serial-port connection: UART for Z8 and COM1 for host PC.
- PWM signal controls the H-bridge to perform the motion control, which changes the speed and the direction of the DC motor.
- The cart is directly assembled with the DC motor. With the cart's movement in desirable, the pole on it will be in the upright position dynamically stable.

1.4 The Format of Report

Chapter 1: An introduction to the project and an overview of the overall systems. A brief format of the report organization is presented.

Chapter 2: ANFIS controller design. Analyze the system and do the simulation in Matlab environment. Build up the ANFIS structure, and then train it using the data gathered from the existing demo in Matlab. The control performance of the generated ANFIS has been displayed.

Chapter 3: Hardware design. Do research in each key component in needed, design the framework of the system, and assemble them into a whole hardware structure. The completed hardware is illustrated in this chapter.

Chapter 4: Configuration of Z8® microcontroller. Assign the GPIO pins for data input from the sensors, set up the Timer as the PWM output, and program the UART for communicating with host PC.

Chapter 5: Implementation and testing. The motion control unit of DC motor and H-bridge has been tested. Also, the connection has been built up between the ACE™ sensor, the Z8 MCU and the Hyper Terminal® in host PC.

Chapter 6: Project management. Describe the time arrangement and explain the schedule alteration. The project costing and required resources are listed.

Chapter 7: Conclusion and further discussion. Summarize the achievements in this project, and comment on the final outcome. The future work and improvement have been presented and discussed.

Chapter 2 ANFIS CONTROLLER DESIGN

2.1 What is ANFIS?

ANFIS: Adaptive-Network-Based Fuzzy Inference System, which is "a fuzzy inference system implemented in the framework of adaptive network", was first introduced by Roger Jang in University of California, Berkeley, 1993. [3]

Using a given input/output data set, ANFIS constructs a fuzzy inference system (FIS) whose membership function parameters are tuned (adjusted) using backpropagation training algorithm. This learning ability allows the fuzzy systems to be modeled and modified just by these training data.

2.1.1 Sugeno Fuzzy Inference System

Fuzzy inference is a process of mapping from given input(s) to an output, using the theory of fuzzy sets.

Fuzzy logic system is based on IF-THEN rules. Sugeno fuzzy rules are expressed in the following form:

Rule 1:

IF x_1 is A_1,

AND x_2 is A_2,

……

AND x_n is A_n,

THEN $y_1 = f_2(x_1, x_2, ...x_n)$

Rule 2:

IF x_1 is B_1,

AND x_2 is B_2,

……

AND x_n is B_n,

THEN $y_2 = f_2(x_1, x_2, ...x_n)$

……Rule N:

IF x_1 is N_1,

AND x_2 is N_2,

……

AND x_n is N_n,

THEN $y_n = f_n(x_1, x_2, ...x_n)$

Where $x_1, x_2, \ldots, x_n$ are input variables; $A_1, A_2, \ldots, A_n$; $B_1, B_2, \ldots\ldots, B_n$; $N_1, N_2, \ldots\ldots, N_n$ are fuzzy set.

When y is a constant, it is a zero-order Sugeno fuzzy model in which the consequent of a rule is specified by a singleton. When y is a first-order polynomial, i.e.

$$y_k = k_0 + k_1 x_1 + k_2 x_2 + \ldots + k_n x_n \tag{2.1}$$

It is called first-order Sugeno fuzzy model.

The output of Sugeno fuzzy inference is the weighted average:

$$y = \frac{\mu_1 y_1 + \mu_2 y_2 + \ldots + \mu_n y_n}{\mu_1 + \mu_2 + \ldots + \mu_n} \tag{2.2}$$

2.1.2 ANFIS Architecture

ANFIS is normally represented by a six-layer feedforward neural network. [1]

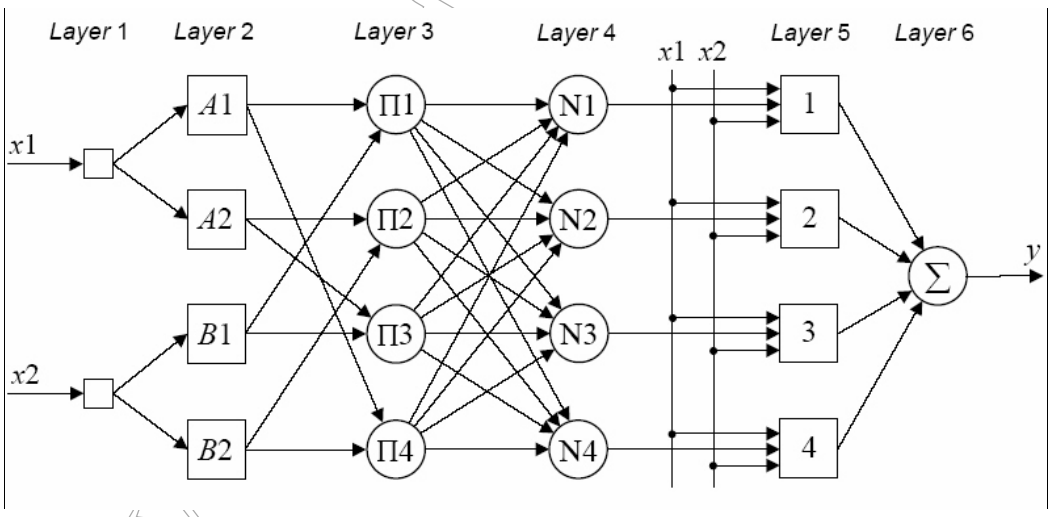


Fig 2.1 Adaptive Neuro-Fuzzy Inference System (ANFIS)

***Layer* 1: Input layer.** Neurons in this layer just pass external crisp signal to *Layer* 2.

***Layer* 2: Fuzzification layer.** Neurons in this layer perform fuzzification.

***Layer* 3: Rule layer.** Each neuron in this layer corresponds to a single Sugeno-type fuzzy rule. A rule neuron receives inputs from the respective fuzzification neurons and calculates the firing strength of the rule it represents.

***Layer* 4: Normalization layer.** Each neuron in this layer receives inputs from all neurons in this

layer receives inputs from all neurons in the rule layer, and calculates the rule layer, and calculates the **normalized firing strength** of a given rule. The normalized firing strength is the ratio of the firing strength of a given rule to the sum of firing strength of a given rule to the sum of firing strengths of all rules. It represents the contribution strengths of all rules. It represents the contribution of a given rule to the final result.

***Layer* 5: Defuzzification layer.** Each neuron in this layer is connected to the respective normalization neuron, and also receives initial inputs.

***Layer* 6: Summation layer.** This neuron calculates the sum of outputs of all defuzzification neurons and produces the overall ANFIS output.

ANFIS uses hybrid learning algorithm: Least-Square Estimator and Gradient Descent method to learn the rule consequent parameters and rule antecedent parameters, and turn the membership functions as well.

2.2 Overview of Control Problems

The brief control principle of an inverted pendulum system is by controlling the motion of the cart, keeping the pole standing upright and dynamically stable.

The cart can only move along the tracks, forward and backward, and the pole is fixed on the pivot shaft on the cart, so both the pole and the cart have one-dimension motion space.

The whole system is controlled by the force (*F*) which is applied onto the cart. The magnitude and the direction of the force are modified in real-time based on the combination of the pole angle and the cart position.

2.3 Design the ANFIS Controller

2.3.1 Training Data

ANFIS has a network-type structure, which maps the inputs through input membership functions and associated parameters to the outputs. This is quite similar to a neural network. So, like any learning algorithm, the most important step to get a desirable ANFIS controller is to collect adequate and proper training data sets.

There are different ways to obtain the training data. A straightforward thought is getting help from an expert, who can manually handle the cart and manage the whole system in stable. Just imagine keeping a pen standing on you finger. But, in practical, it is not easy to find such “experts”, so taking some help from the existing successful controllers can benefit our work much easier and quicker.

2.3.2 ANFIS Structure

Output and Input:

For an inverted pendulum system, there are in total four inputs:

θ (rad): The angle of the pendulum with respect to the absolute zero, upright. θ is positive while the pendulum is leaning to right, and negative to left.

$\dot{\theta}$ (rad/s): The angle velocity of the pendulum. $\dot{\theta}$ is positive if the pendulum is swinging clockwise, and negative anticlockwise.

x (cm): The position of the cart with respect to the absolute zero, center of the tracks. x is positive when the cart is on the right side of the tracks, and negative on the left.

$\dot{x}$ (cm/s): The moving speed of the cart. $\dot{x}$ is positive when the cart is moving to right, and negative to left.

and one output:

F (N): The force applied on the cart. F is positive means the force is drawing the cart to right, while negative to left.

Here, questions may arise: Do we need all the 4 inputs? Is the cart position necessary for keeping the pendulum upright? Can we simply control the system only taking account of the pole angle?

Just imagining when we are trying to keep a chopstick standing upright on you finger, it seems that we only focus on the position of the chopstick, rather then the position of your finger. And the control rules also look like very simple: if the chopstick is falling down to the left, move your finger to the left; if it is falling down to the right, move your finger to the right.

But, think about a particular situation: both the θ and the $\dot{\theta}$ are zero. In this case, the pole is exactly upright, and no movement on the pivot. So, definitely, the output F should be zero, no force applied. But, without any controlling force, the cart will keep in moving due to the inertia and momentum, and at the end crash in hitting the edges of the tracks.

It may be proposed that why not change the output from *Force* to *Speed* (the speed to cart). And revise the rules as follow:
If the pole is falling to left, then *Speed* is negative (to left).
If the pole is falling to right, then *Speed* is positive (to right).

If the pole is upright, then *Speed* is zero.

This seems feasible, but in the real hardware system, the output *Speed* from controller cannot act directly onto the cart, which still has to be converted into force. Their relationship can be described in the following equation:

$$F = m\frac{dv}{dt} \tag{2.3}$$

Where v is the *Speed*, and m is the mass of the cart.

Actually, in reality, the cart speed can not jump from one value to another instantaneously, otherwise F will be infinite large. So this supposition is impossible to be realized and implemented.

Finally, it has been confirmed that all the 4 inputs are compulsory.

Membership Functions:

Based on "the simpler, the better" of any neuro-network based structure, first, we set 3 membership functions (MFs) for each input, as well as the output. So there are 3^4=81 mapping rules in total, adequate to resolve this control problem.

The ANFIS structure is illustrated below:

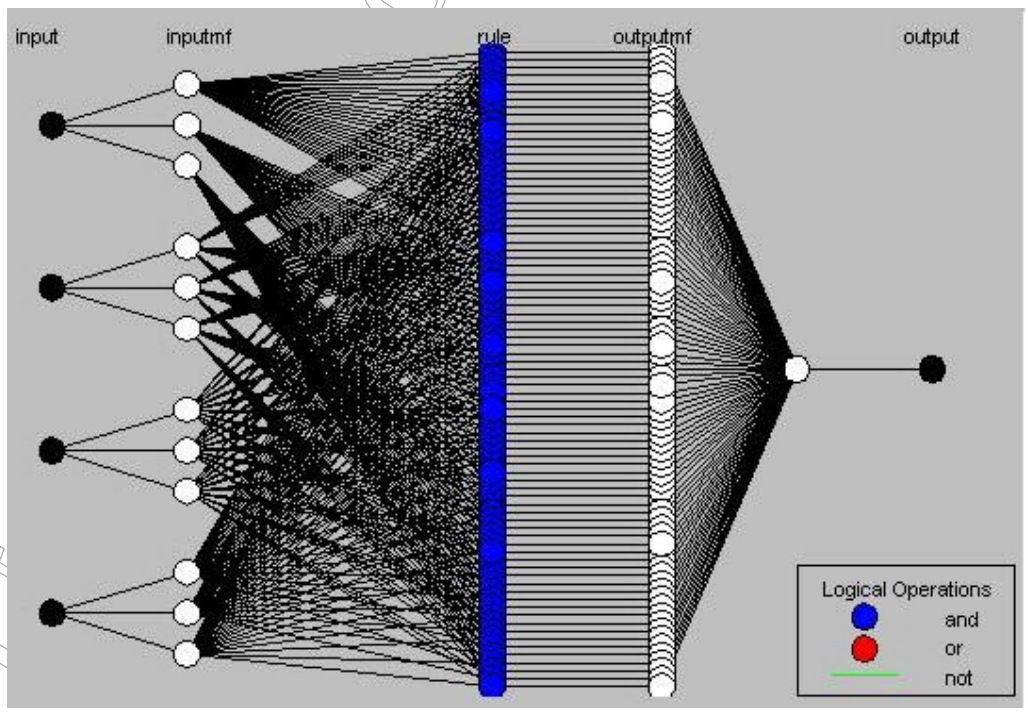


Fig 2.2 ANFIS structure

2.4 Simulation in Matlab

2.4.1 System Modeling

Although using ANFIS, we can get rid of the bothersome job in calculating the complicated mathematic equations of the inverted pendulum system, even no need to know the correlation of all the parameters. But to do the simulation, still we need to understand these equations in order to build up an ideal system model.

State equations of inverted pendulum:

$$\ddot{\theta} = \frac{g\sin(\theta) + \cos(\theta)\left(\frac{-F - mL(\dot{\theta})^2\sin(\theta)}{m_c + m}\right)}{L\left(\frac{4}{3} - \frac{m\cos^2(\theta)}{m_c + m}\right)} \tag{2.4}$$

$$\ddot{x} = \frac{\{F + mL(\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta)\}}{m_c + m} \tag{2.5}$$

Where: g : acceleration due to gravity. (9.8m/sec^2)

θ : angle of pole in *rad*

x : horizontal position of cart in *m*

m_c : mass of cart in *Kg*

m : mass of pole in *Kg*

L : the distance from the pivot to the pole's center of mass in *m*

F : applied force in *Newton*

In Simulink®, a demo of an inverted pendulum has existed in Fuzzy Logic Toolbox, which is called "Cart and Pole". In this demo, the controller is a Fuzzy Logic Controller using FIS (Fuzzy Inference System). The block diagram of the whole system in Simulink ("slcp") is shown in *Fig 2.3*.

2.4.2 Generate ANFIS Controller

We can consider the Fuzzy Logic Controller in this demonstration as an "expert" who has succeeded keeping the cart and pole system in balance. In this way, using the "*To Workspace*" function block from Simulink library, we can easily obtain the input and output data sets, and store them in the *Workspace*. Then we can compose them into a 5-dimensional array, where the last column must be the output. (*Fig 2.4*)

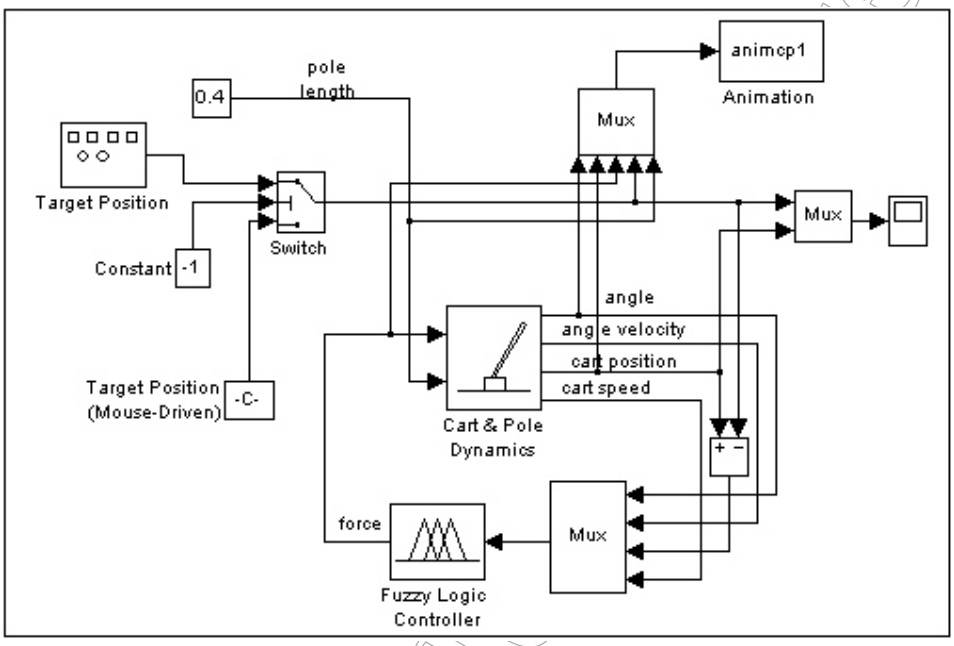


Fig 2.3 Simulation system in Matlab

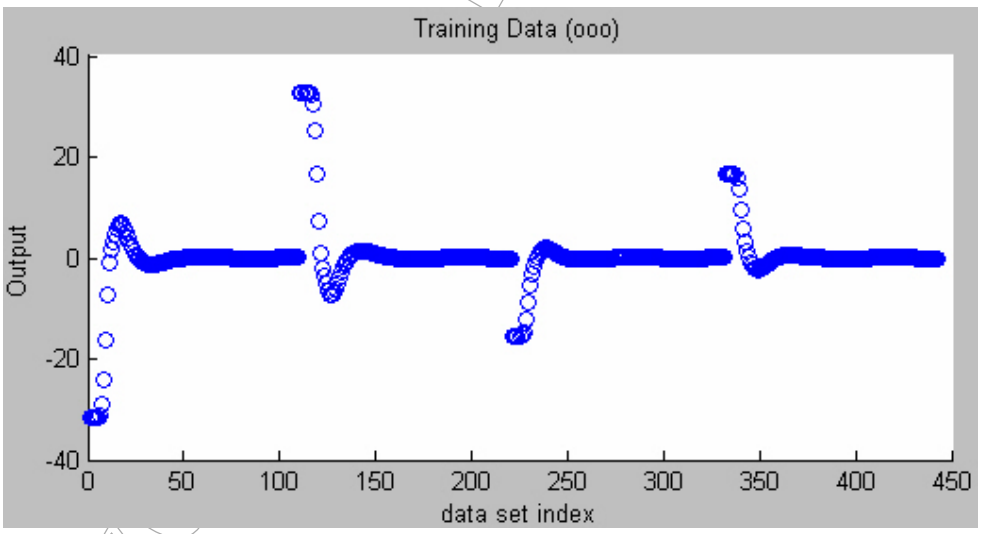


Fig 2.4 Training Data Set Index

To generate FIS, the parameters of Train FIS are set as:

Grid Partition: partitioning method;

Optimize Method: hybrid;

Error Tolerance: 0;

Training Epochs: 20;

Generate FIS menu is set as follow: (*Fig 2.5*) (refer to *Fig 2.2* for the generated ANFIS structure)

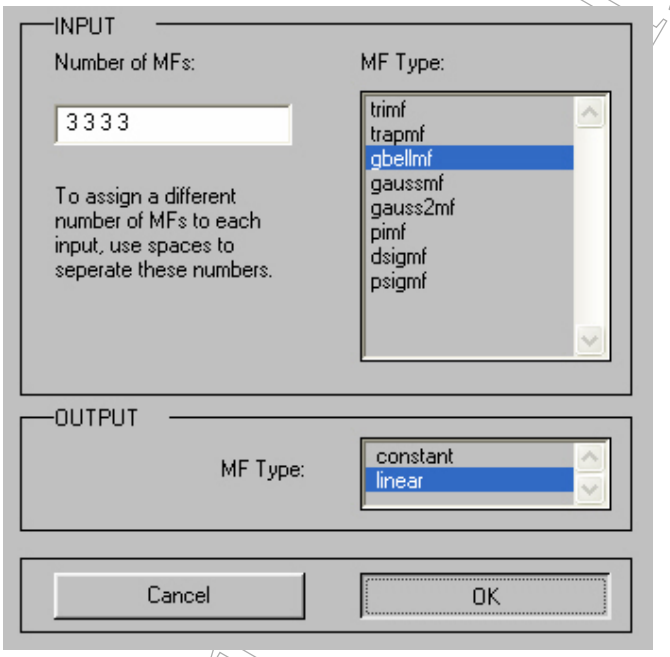


Fig 2.5 Setting of Generate FIS

Using these data sets as the training data, the ANFIS controller can be generated from "ANFIS Editor" GUI. But here, the most important factor that decides whether the generated controller is desired or not, is how well the training data are.

To any neuro-network based system, optimizing the training data is always a big problem. Only a few data may be not enough to train the system properly, while too many may cause over-training. (Just like in my simulation, the first ANFIS controller, which was generated by 1328 data sets, worked quite well for all occasions, but failed when the initial pole angle was 0.2rad.) Finally, 443 data sets are chosen as the training data. In addition, the initial condition of the cart and pole dynamics has been experimented several times with different parameter settings, so that it benefits the training procedure because the more comprehensive and appropriate training data are collected.

2.4.3 Performance of ANFIS Controller

The generated ANFIS controller has been tested by applying it into the demo to replace the Fuzzy Logic Controller. The controlling result is displayed in *Fig 2.6*: (pole length: 0.3 meter; and the initial pole angle is set to 0.1rad, all others are zero.)

Then, the value of the initial pole angle is increased by 0.1rad each time. The biggest initial pole angle this controller can manage is 0.8rad, (= 45.86°), with a similar result as above. But, when

the initial angle is 0.9rad, it fails. (*Fig 2.7*)

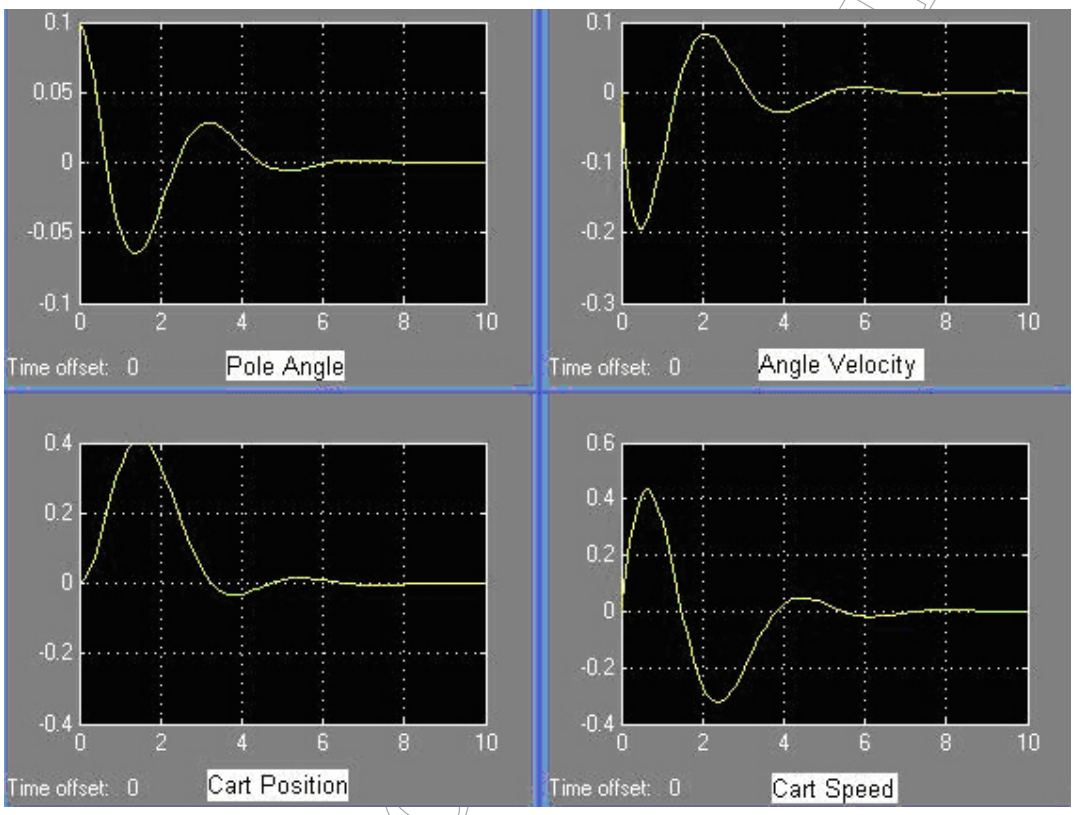


Fig 2.6 Control result in simulation (pole angle: 0.1rad)

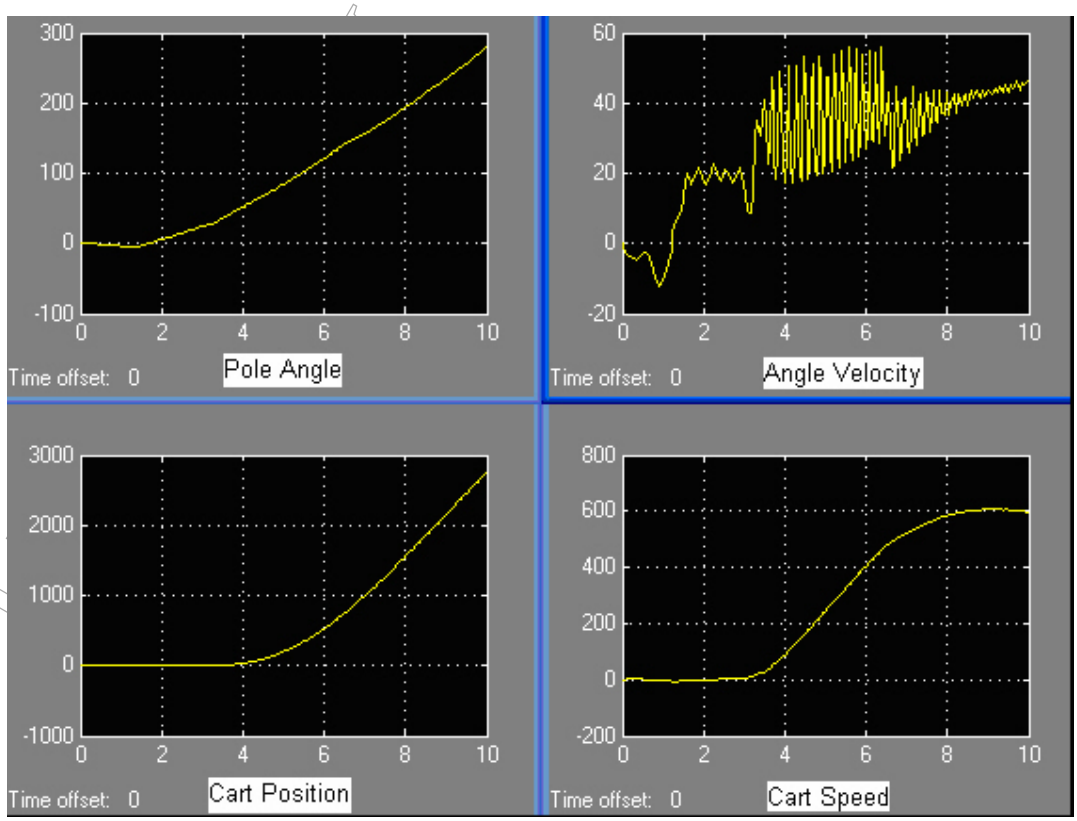


Fig 2.7 Control result in simulation (pole angle: 0.8rad)

Then the pole length is increased by 0.1m each time from 0.3m to 0.8m. With the length rising, the maximum of the initial pole angle is reduced. But amazingly, the exception is if the pole length is cut to 0.1m, the maximum initial pole angel is only 0.18rad! *Table 2.1* shows the simulation result.

Pole Length (meter)	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.8*	1.0
Maximum of Initial Pole Angle (Rad)	0.17	0.18	0.18	0.83	0.73	0.67	0.60	0.11	NA

Table 2.1 Result of the maximum initial pole angle of different pole length

When the pole is longer then 0.8m, the ANFIS controller can never keep the system in stable; even the initial pole angle is 0.

*: When the pole length is set to 0.8m, system has another type of balancing status. That is the pole is standing on the cart without falling down, but the cart is vibrating fast in a certain distance on the tracks, instead of staying in the center. (shown in *Fig 2.8*)

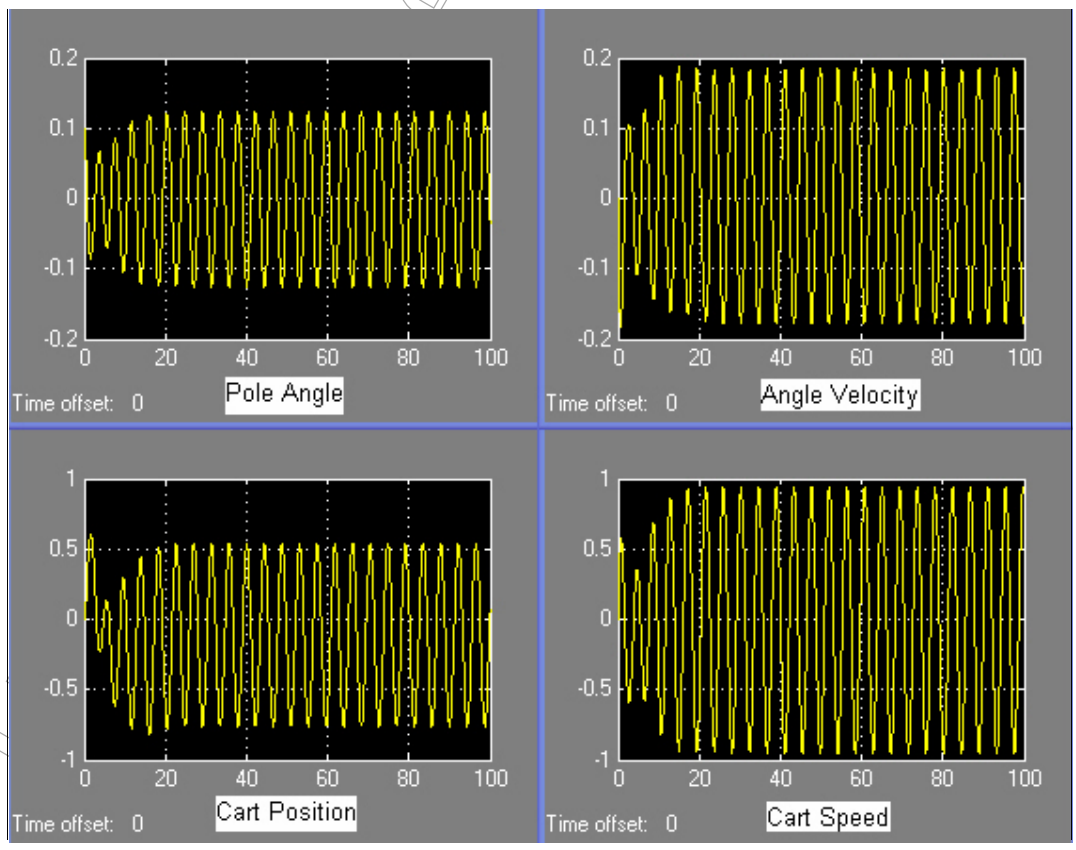


Fig 2.8 Balancing status at pole length: 0.8m

In generally, this ANFIS controller has a nice control performance when the pole length is in the range of 0.3m to 0.6m.

2.4.4 FIS of Two Membership Functions

But, checking the demo ("slcp") more carefully in Simulink®, it was found that the Fuzzy Logic Controller only has two membership functions (2-MF) for each input variable in FIS, and therefore only 16 (2^4) rules! It is evidenced that a 2-MF FIS is competent for the ANFIS controller design.

Motivated by this, another ANFIS, with 2-MF FIS, is generated by the same training data. (Follow the same steps as previous.)

Amazingly, the result of this ANFIS controller is even better then that one of 3-MF. The maximum initial pole angle is 1.1rad (=63°), which means the pole can be handled in the range of around 120 degrees! (*Fig 2.9*, pole angle: 1.1rad)

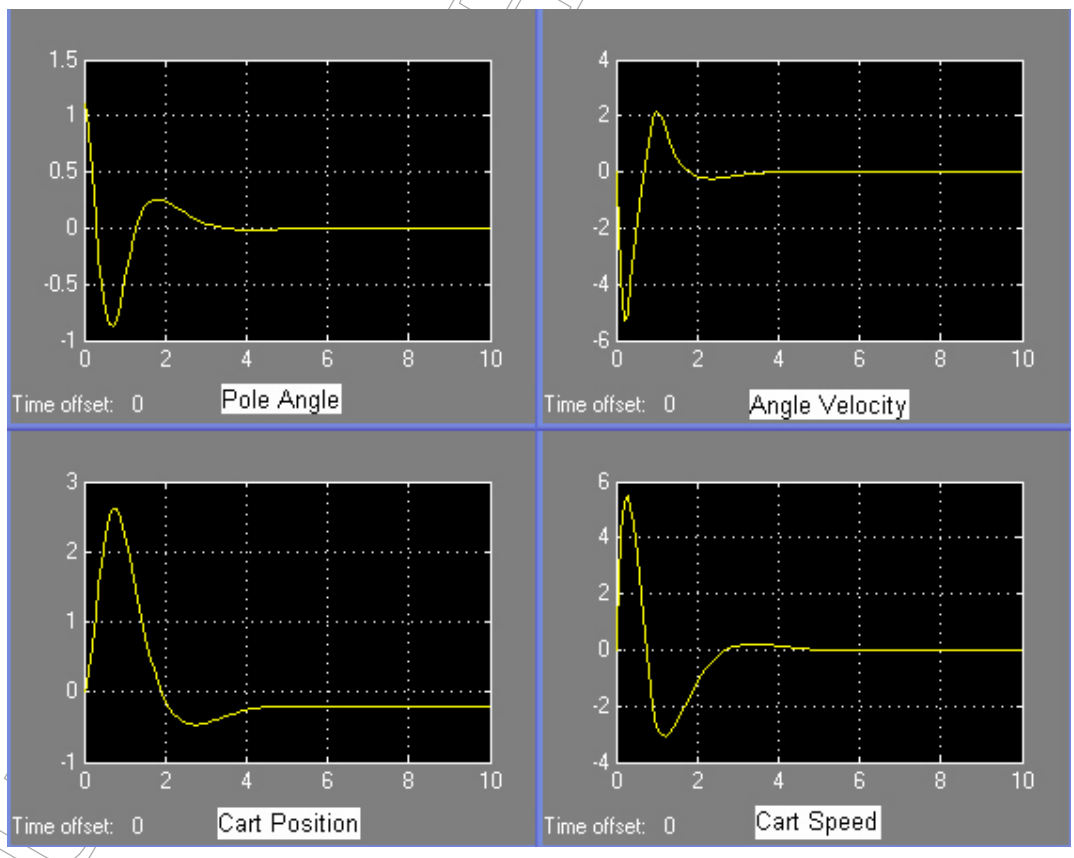


Fig 2.9 Simulation result of 2-MF ANFIS controller

It is not easy to explain the reasons on the performance of two different ANFIS controllers, because the training procedure is invisible. The potential reason is that the 3-MF FIS has a more complicated structure, which may be more specified for particular circumstance, but less robust.

2.5 Chapter Summary

This chapter has introduced the theory of ANFIS, also briefly explained the structure and its self-learning process. Then focusing on this application, the ANFIS controller for an inverted pendulum system has been designed and some issues on it have been discussed. Finally, the ideal model is built up in Matlab Simulink environment, and the ANFIS controller is generated from the training data and tested in the demo. The results have illustrated that in simulation, this ANFIS controller is competent in keeping the cart and pole system stable and balance.

Chapter 3 HARDWARE DESIGN

3.1 Framework Design

3.1.1 Cart

Cart design is a tradeoff between the agility and the stability. A very small and light cart can be easily and quickly steered with less inertia, but the stableness is not good enough, especially when the pole is swinging.

At the beginning, it was intended to use Lego® kit to build up the framework of this cart and pole system. Lego toy comes with varied of bits with different sizes and shapes. Using these parts almost any required models can be constructed. The Lego cart meets some of the requirements very well, like light weight and easy build, but due to the plastic material, the big problem is the weakness in solidity and strength. Also, it is quite hard to assemble a normal DC motor with the Lego parts, unless using Lego DC motor from the kit. After trying several different patterns of Lego, it was abandoned.

So, designing and manufacturing a metal cart is the last but best solution. (See *Appendix A* for mechanical drawing.) The 4 wheels and 2 axles are selected from Lego modules.

3.1.2 Shaft

The shaft, which the pendulum is attached to, need more special consideration. The shaft should be firmly fixed on the cart, no axial movement, but able to rotate freely, with less friction. So, two flanged bearings are designed to hold the shaft substantially with the cart.

Furthermore, there must be strict no movement between the pendulum and the shaft, because the pendulum angle is measured by the optical encoder whose codewheel is fixed on the shaft. Accordingly, a bolt with nut has been lathed in one end of the shaft to grip the pole in stationary. (See *Appendix A* for mechanical drawing.)

3.1.3 Driving Set

The main driving device is consisted of a motor, an idle pulley and a belt. In order to avoid any slippage between the driving wheel and the belt, the best choice is a set of timing belt and pulley, which have the gear teeth in mesh.

But, unfortunately, the timing belt is normally designed for heavy load applications in industry. It is not easy to purchase a cheap, slim, long and flexible timing belt suitable for this inverted pendulum system. At last, a normal nylon string has been adopted as an alternative, and the

pulley is replaced by a small plastic wheel which is glued together with one gear-wheel from the gearbox.

3.2 Sensors

To get all the 4 inputs, two sensors are needed: one for pendulum and one for cart. The principles of sensor selection are based on capacity, feasibility, reliability and price.

3.2.1 Optical Encoder

For pendulum, the angle of the pole with respect to the absolute zero (upright) is measured. A few types of sensors can provide the motion detection, such as potentiometer, optical shaft encoder, incremental quadrature encoder, etc. To prevent introducing any extra friction onto the shaft, the optical encoder with codewheel is the best choice, because these devices make absolutely no mechanical contact. After studying and comparing several different types of optical encoder in the market, the following selections are made:

- HP HEDS-9140: three channel optical incremental encoder module
- HP HEDS-5140: three channel codewheel

The main features of HP HEDS-9140 are listed below: (from datasheet)

- Two channel quadrature output with index pulse
- Resolution up to 2000 CPR (Counts Per Revolution)
- Low cost
- Easy to mount
- No signal adjustment required
- Small size
- -40°C to 100°C operating temperature
- TTL compatible
- Single 5 V supply

The encoder module consists of a lensed LED (Light Emitting Diode) as the light source and a signal processing IC as the light detector. (Fig 3.1)

We can see from the block diagram above that the parallel light beam is interrupted by the pattern of space and on the codewheel which rotates between the emitter and detector. Simultaneously, these interruptions detected by photo-diodes are arranged in a pattern that corresponds to the radius and design of the codewheel.

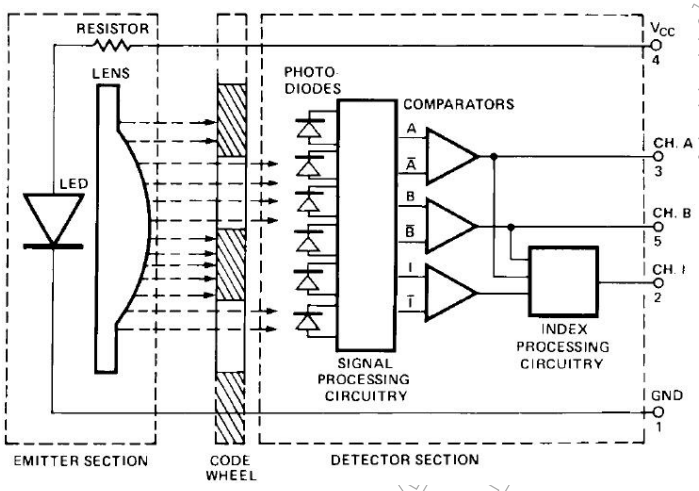


Fig 3.1 Optical encoder

Output of optical encoder: (*Fig 3.2*)

Channel A and B: The final outputs are yielded from the IC comparators which process the signals generated by the photodiode. The digital output of channel A is in 90 degrees out of phase with that of channel B, which is called as "quadrature".

Channel I: Index signal. A pulse is generated once for each full rotation of the codewheel.

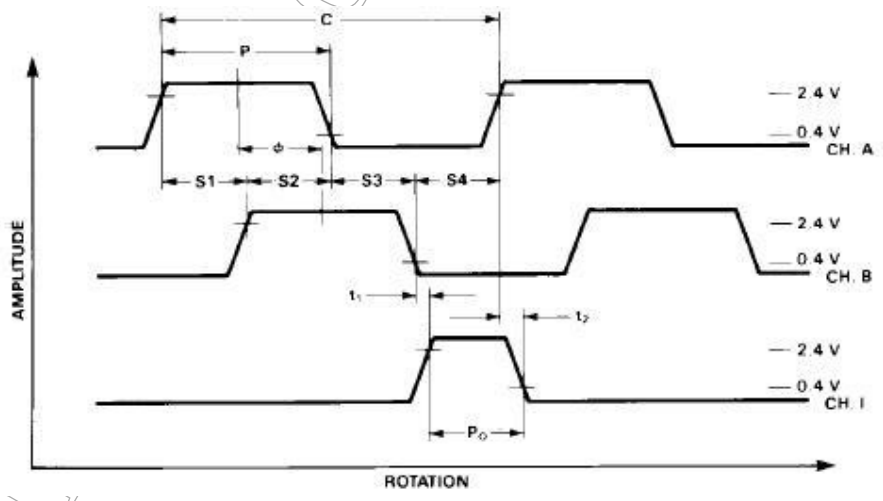


Fig 3.2 Output waveforms

The rotation direction can be derived by checking the signals that which channel is leading the other, and the position of pendulum can be derived by counting the pulses in a certain direction. In this method, an initial position (absolute ZERO) needs to be set with all movements being related to this position, which is known as "Incremental Encoding".

3.2.2 Absolute Contacting Encoder (ACE^{TM})

To measure the position of the cart on the track, another sensor is required. Comparing several possible sensors in the market, this Absolute Contacting Encoder (ACE^{TM}) has been chosen. It is adequate in this application, but with the lowest price.

The Absolute Contacting Encoder (*Fig 3.3*) is actually using gray coding method, which ensures there is only one bit changing at each step. A binary "1" denotes an "open" switch while a binary "0" denotes a "close" switch. The 8-bit digital output gives 128 states of the shaft position, and position 0-127 are seen by a clockwise rotation of the shaft. (Refer to *Appendix B* for the correlation of the output data to the corresponding actual position)

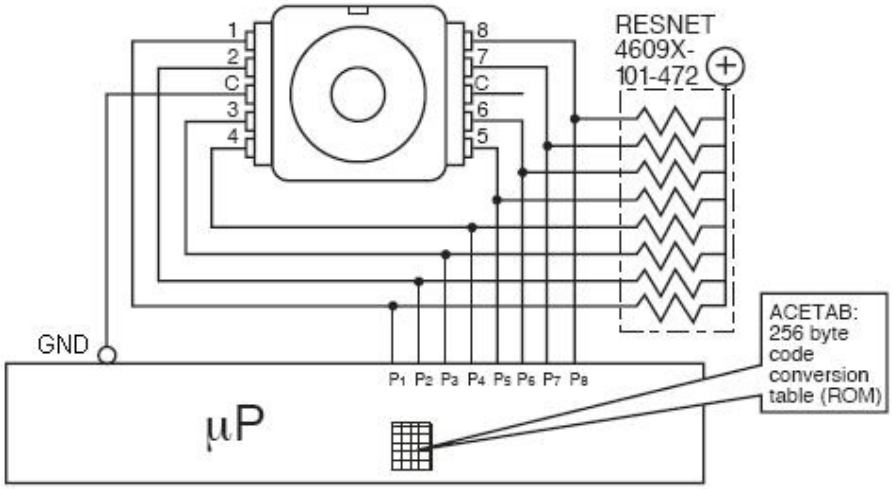


Fig 3.3 Absolute Contacting Encoder (ACE^{TM})

Remarkably, one resistance net (RESNET in *Fig 3.3*) must be added between the ACE and Z8 MCU; otherwise, the sensor may be damaged.

To get the accurate position of the cart, the movement of this ACE sensor has to be restricted in ONE revolution. In order to match the whole distance of the cart moving on the tracks into one revolution of ACE sensor's shaft, a gearbox has to be equipped to reduce the rotating velocity. The gear-down ratio is 16:1, using two gear wheels. (The gearbox is taken out from the Motor & Multi-Ratio Gearboxes set, available in store.)

3.3 Microcontroller (MC)

In this application, the microcontroller works as an interface between the host PC/ANFIS controller and the hardware components. The requirement of the microcontroller is that it can read the digital signals from the sensors and send them to host PC as input data, and then according to the output data from the PC/controller, generates the PWM signal for motion control.

Various microcontrollers can be used in this application. The Z8 Encore!® microcontroller has been selected, which is integrated onto the development board, because it is currently available in the department, and also it has been used in a similar final project last year. (*Fig 3.4*)

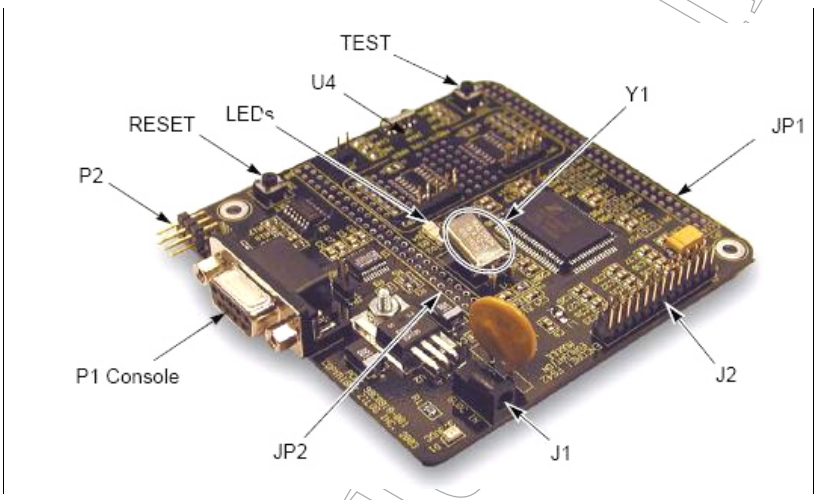


Fig 3.4 Z8 Encore!® 64K serial MCU development board

The Z8 Encore!® Z8F642 MCU, which is a member of ZiLOG microcontroller products based on the 8-bit eZ8 core CPU, is equipped on this development board.

The main features of the Z8 Encore! Z8F642 are listed below: (from User Manual)

- 64K of Flash memory with in-circuit programming capability
- 4K of register RAM
- Either eight or twelve channels, 10-bit analog-to-digital converter (ADC)
- Two Full-duplex UART
- Serial Peripheral Interface (SPI)
- 3 or 4 16-bit timers with capture, compare, and PWM capability
- Watch-Dog Timer (WDT) with internal RC oscillator
- Up to 60 I/O pins
- Programmable priority interrupts
- On-Chip Debugger
- Power-On Reset (POR)
- 2.7–3.6V operating voltage with 5V-tolerant inputs
- Operating temperature: 20° ± 10°C

After checking the datasheet of this microcontroller development board, it has been confirmed this product meets well all the requirements in this inverted pendulum application. Also, previous applications have proved Z8 is suitable for motion control.

3.4 Motion Control Unit

3.4.1 DC Motor

At the beginning, I intended to use a servo motor as the controller of the cart motion. But a servo motor can only work in one revolution, which is not enough to cover the whole length of the track. Also, in general, the price of a servo motor is a bit higher then a normal DC motor. Considering the speed of the DC motor, which is much faster then the required velocity, a normal DC motor with a gearbox is preferred.

The features of the DC motor with a gearbox are listed below: (from product specification)

- Each pinion-to-gear ratio is 4:1
- Operation voltage: 1.5 to 3.0 V
- Current consumption range: 0.2 to 0.8 A

By simulating the inverted pendulum system in Matlab, we can find that the highest speed needed in controlling the cart is no more then 0.5 m/s, so 2 gears are assembled together to gear down the motor speed (16:1, approximate 845 r.p.m) into a reasonable and adequate speed with the maximum torque.

3.4.2 H-Bridge

To use a normal DC motor for motion control, the average speed and the running direction of the motor should be able to be changed. The direction is decided by the polarity applied to its two terminals. The widely applied method to achieve this current change is H-bridge. The basic H-bridge is shown in *Fig 3.5*

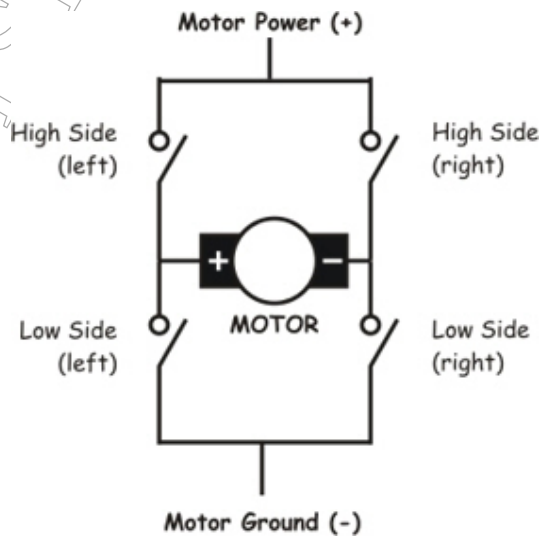


Fig 3.5 Basic H-bridge

H-bridge, also called full-bridge, is a circuit with 4 switches, and the current flow, which controls the motor forward or reverse, is decided by which switches are turned on. The 4 states are shown below: (*Table 3.1*)

High Side Left	High Side Right	Lower Left	Lower Right	State Description
On	Off	Off	On	Motor goes Clockwise
Off	On	On	Off	Motor goes Counter-clockwise
On	On	Off	Off	Motor "brakes" and decelerates
Off	Off	On	On	Motor "brakes" and decelerates

Table 3.1 Four states of H-Bridge

Initially, it was planned to use 2 PNP, 2 NPN transistors and 4 diodes to build up my own H-bridge. But, after trying, I found it was not so easy to make up a reliable full-bridge circuit, so the LMD18201 H-bridge was bought. (*Fig 3.6*)

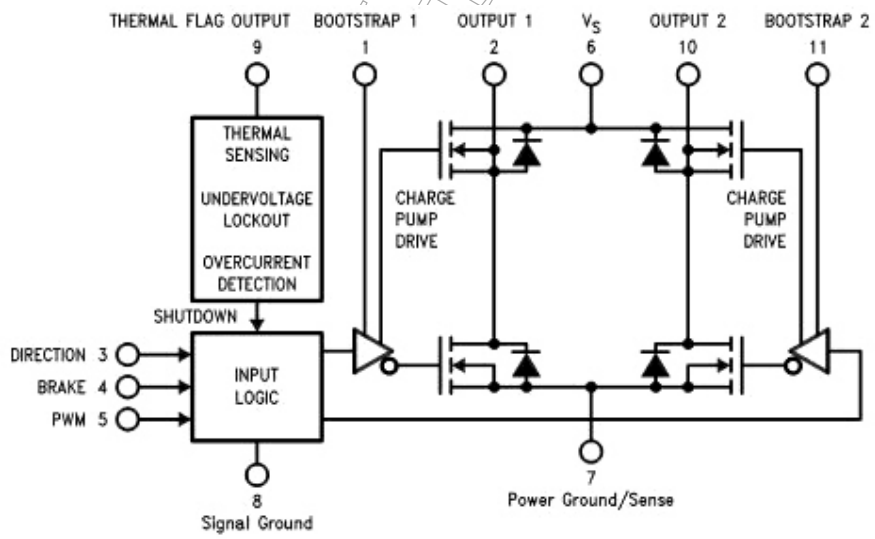


Fig 3.6 Functional block diagram of H-bridge

The main features of LMD18201 are listed below: (from datasheet)

- Delivers up to 3A continuous output
- Operates at supply voltages up to 55V
- TTL and CMOS compatible inputs
- Thermal warning flag output at 145°C
- Thermal shutdown (outputs off) at 170°C
- Internal clamp diodes
- Shorted load protection

But the minimal operation voltage of this H-bridge is +12V, which is much higher then the working voltage of the DC motor (1.5-3.0V). In order to protect the DC motor from burning-up,

a 50Ω thermal-cut (heat-sensitive) resistor is added between the H-bridge PWM signal pin and one of the DC motor poles. We select 50Ω resistor because the working resistance of DC motor is around 15Ω, so the supplied voltage to the DC motor is:

$$V_{DC-motor} = 12V \times \frac{15\Omega}{(15+50)\Omega} = 2.77V \quad (3.1)$$

which is perfect in the range of the DC motor working voltage.

3.5 Complete Hardware System

The overall hardware system is illustrated in the following photos:

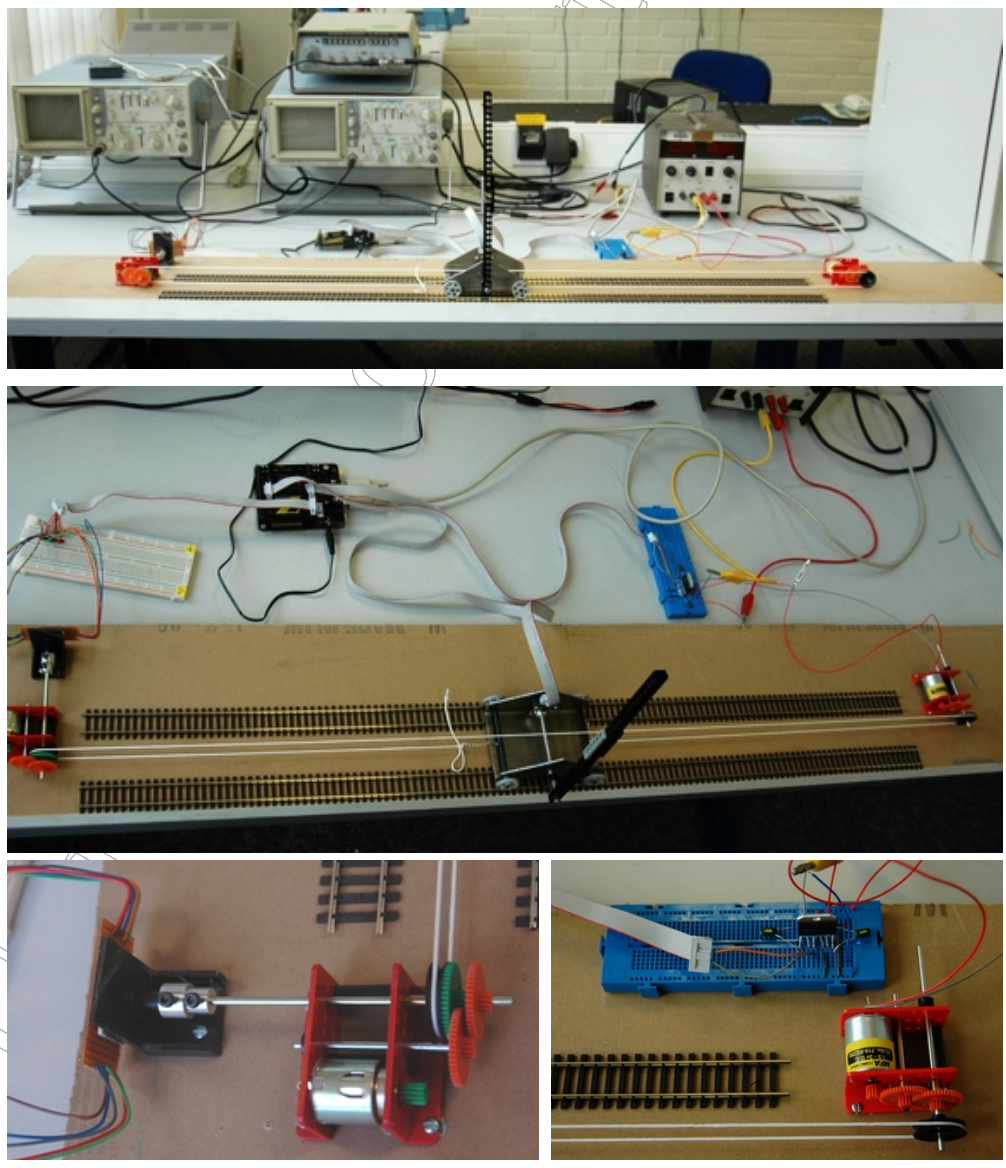


Fig 3.7 Photo of complete hardware system

And the complete system circuit connection is shown below: (*Fig 3.8*)

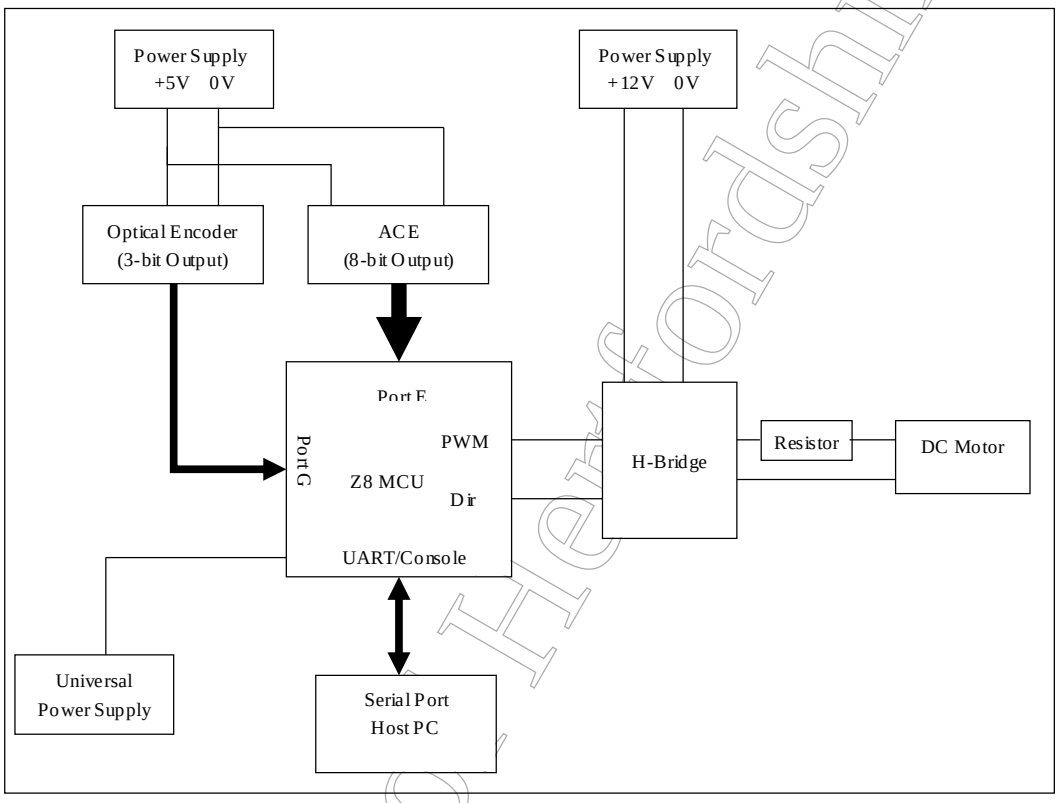


Fig 3.8 Complete system circuit connection

3.6 Chapter Summary

This chapter has described and illustrated all the important parts of the inverted pendulum hardware system. The reasons of each decision of the component selection have been explained. Also some problems in assembling and fixing are considered and resolved. Finally, the whole hardware structure of an inverted pendulum system has been completed and ready for testing and configuration.

Chapter 4 CONFIGURATION OF Z8 MICROCONTROLLER

4.1 Configuration of GPIO

GPIO stands for General Purpose Input/Output port. In Z8 microcontroller, there are in total seven 8-bit ports (Ports A-G) and one 4-bit port (Port H) for general-purpose input/output (I/O) operations.

In my application, 11 pins are needed as input port to get data from the sensors: 8 pins for ACE and 3 pins for optical encoder. Meanwhile, 2 pins are assigned for output signals PWM and Direction. Actually, most of the pins on board can be configured as input/output, but Port E (pin [7:0]) has been selected for ACE, Port G (pin [1, 2, 3]) for optical encoder, and Port C (pin [1]) for PWM (see Chapter 4.2), Port D (pin [3]) for Direction. (Refer to Schematic of Z8 MCU development board in *Appendix C*)

The Registers for each Port providing access to GPIO control, input data and output data are listed below: (Table 4.1 (from manual))

Port Register Mnemonic	**Port Register Name**
PxADDR	Port A-H Address Register (Selects sub-registers)
PxCTL	Port A-H Control Register (Provides access to sub-registers)
PxIN	Port A-H Input Data Register
PxOUT	Port A-H Output Data Register
Port Sub-Register Mnemonic	**Port Register Name**
PxDD	Data Direction
PxAF	Alternate Function
PxOC	Output Control (Open-Drain)
PxDE	High Drive Enable
PxSMRE	STOP Mode Recovery Source Enable

Table 4.1 GPIO Port Registers and Sub-Registers

As input, the received values from port pins are stored in the corresponding Input Data Registers for further processing. And as output, data from the MCU is sent into the Output Data Register, and then transmitted thought these pins to the connected devices/circuits.

4.2 Configuration of PWM

4.2.1 PWM

PWM, Pulse Width Modulated, is a technique that the digital output from a single GPIO pin sets the pulse width of the signal. The pulses have fixed frequency and magnitude, but the pulse width is modulated to represent different analogue level.

The power supplied to the DC motor is switched on and off rapidly according to PWM signal, so the motor speed is decided by the average current from the H-bridge. By changing the register's value of the counter in Z8, the duty cycle of the output varies and so the average DC current changes.

4.2.2 Configuration of PWM Mode

In Z8, the Timers (16-bit up-counters) (*Fig 4.1*) can be configured as PWM mode. Here, Time_1 has been chosen as the PWM generator. The PWM signal is output thought GPIO Port C, Pin1 (PC1_T1OUT).

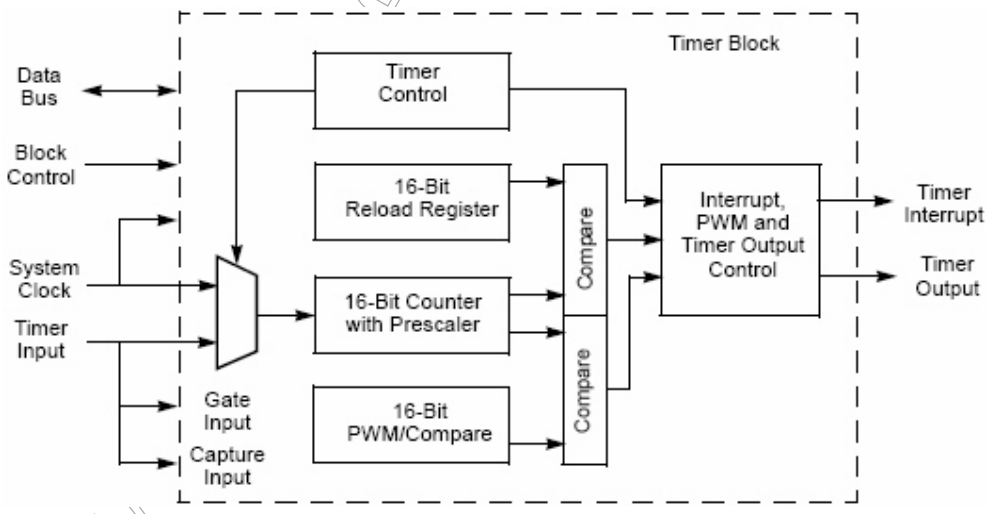


Fig 4.1 Architecture of Z8 Encore! Timer

PWM mode: (*Fig 4.2* shows the Timer_1 Control 1 Register)

0 = Timer Output is forced Low (0) when the timer is disabled. When enabled, the Timer Output is forced High (1) upon PWM count match and forced Low (0) upon Reload.

1 = Timer Output is forced High (1) when the timer is disabled. When enabled, the Timer Output is forced Low (0) upon PWM count match and forced High (1) upon Reload.

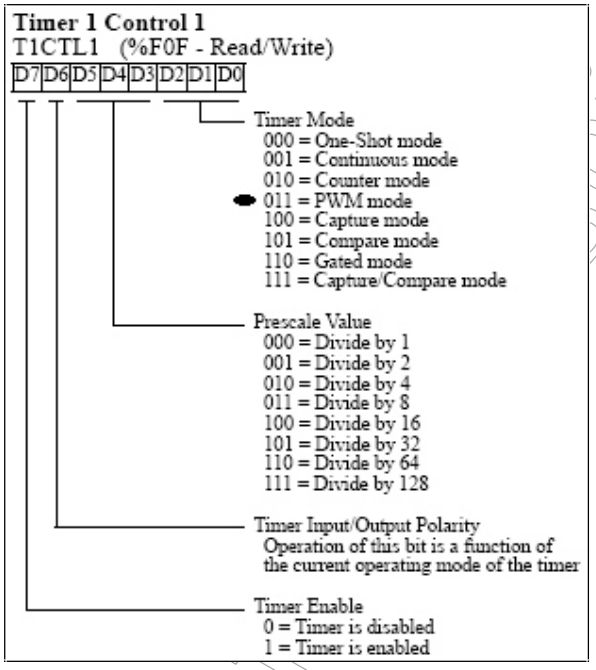


Fig 4.2 Tmer1 Control 1 Register

The Timer_1 PWM High and Low Byte (T1PWMH and T1PWML) registers are being defined, and these two bytes, {PWMH[7:0], PWML[7:0]}, form a 16-bit value that is compared to the current 16-bit timer count. When a match occurs, the PWM output changes state. The PWM output value is set by the TPOL bit in the Timer_1 Control 1 Register (T1CTL1).

The PWM period is determined by the following equation:

$$\text{PWM period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{System Clock Frequency (Hz)}} \tag{4.1}$$

The desired frequency of the PWM signal is 100Hz, (PWM period: 10ms), and Prescale is set to 8. The system clock frequency of Z8F642 is 18.432MHz, so the Reload Value is calculated to be 0x5A00 (Hex).

The TPOL is set to 1, so the ratio of the PWM Output High Time to the total period is calculated as follows:

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{PWM Value} \times 100}{\text{Reload Value}} \tag{4.2}$$

So, changing the PWM Value in Timer PWM Byte Register (T1PWMH and T1PWML), which decides the duty cycle of the modulated PWM pulse, can change the motor speed.

The flowchart of Z8 Timer configuration is illustrated in *Appendix D*. [13]

4.3 Configuration of UART

4.3.1 UART

UART, standing for Universal Asynchronous Receiver/Transmitter, is a full-duplex communication channel capable of handling asynchronous data transfers. UART is configured to build up a connection between the microcontroller and the host PC, for the purpose to communicate the system hardware part with the software part in real time.

The UART always transmits and receives data in an 8-bit data format, least-significant bit first. The data format (without parity) is shown below: (*Fig 4.3*)

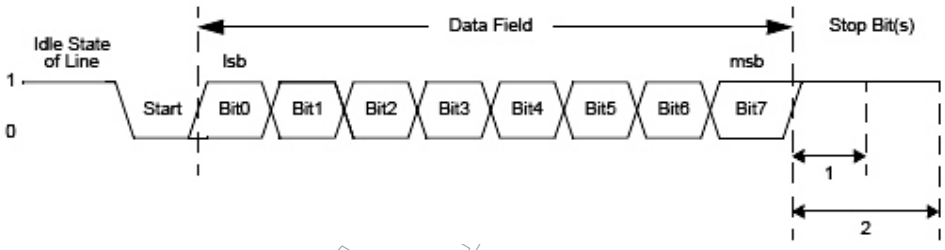


Fig 4.3 UART data format (without parity)

Z8 provides two full-duplex 9-bit UARTs (UART0 & UART1) with bus transceiver Driver Enable control. UART0 has been selected for data transmission. (*Fig 4.4*) The data byte is transmitted to be shifted out through the TXD0 (pin PA5_TXD0).

4.3.2 Configuring UART

Steps to configure UART0 to transmit data using the Polled Method: (modified from [9])

1. Write to the UART0 Baud Rate High and Low Byte registers to set the desired baud rate.
2. Enable the UART0 pin functions by configuring the associated GPIO Port pins for alternate function operation. (pin PA5_TXD0 for data transmission)
3. Write to the UART0 Control 0 register to:
   - Set the transmit enable bit (TEN) to enable the UART0 for data transmission
   - If parity is desired and multiprocessor mode is not enabled, set the parity enable bit (PEN) and select either even or odd parity (PSEL).
   - Set or clear the CTSE bit to enable or disable control from the remote receiver using the CTS pin.
4. Check the TDRE bit in the UART0 Status 0 register to determine if the Transmit Data register is empty (indicated by a 1). If empty, continue to Step 6. If the Transmit Data register is full (indicated by a 0), continue to monitor the TDRE bit until the Transmit Data register becomes available to receive new data.
5. Write the UART0 Control 1 register to select the outgoing address bit.
6. Write the data byte to the UART0 Transmit Data register. The transmitter automatically

transfers the data to the Transmit Shift register and transmits the data.

7. To transmit additional bytes, return to Step 5.

But, even many efforts have been made in executing all above, some problems are still remaining. Finally, it was decided to use the sample code available from ZiLOG documentation.

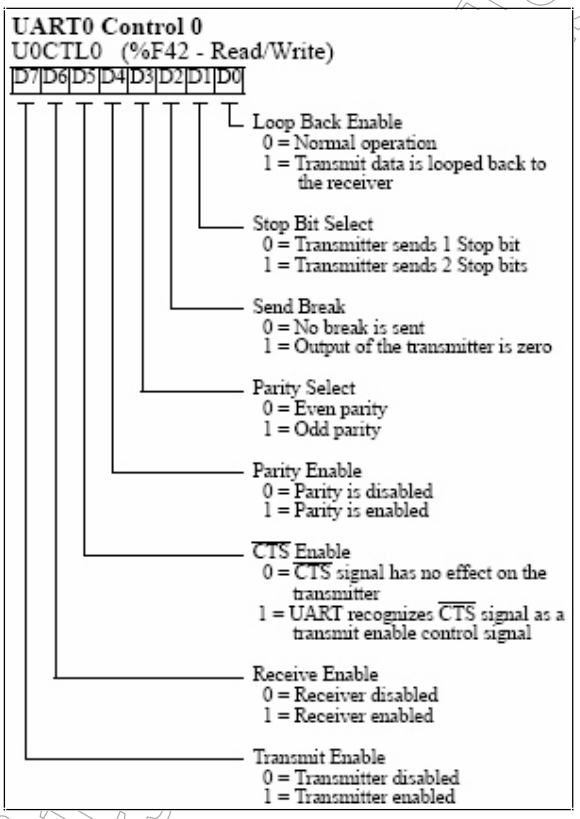


Fig 4.4 UART0 Control 0 Register

4.4 Chapter Summary

In this chapter, the configuration of Z8 MCU has been described. The GPIO is configured for reading data from the sensors and sending control signals as output. Timer is configured as PWM signal generator; and UART for serial port communication with the host PC. After this configuration, the Z8 MCU is capable to work as the interface between the hardware system and the ANFIS controller in host PC.

Chapter 5 IMPLEMENTATION AND TESTING

5.1 Motion Control Unit

The basic motor control circuit using H-bridge is shown below: (*Fig 5.1*)

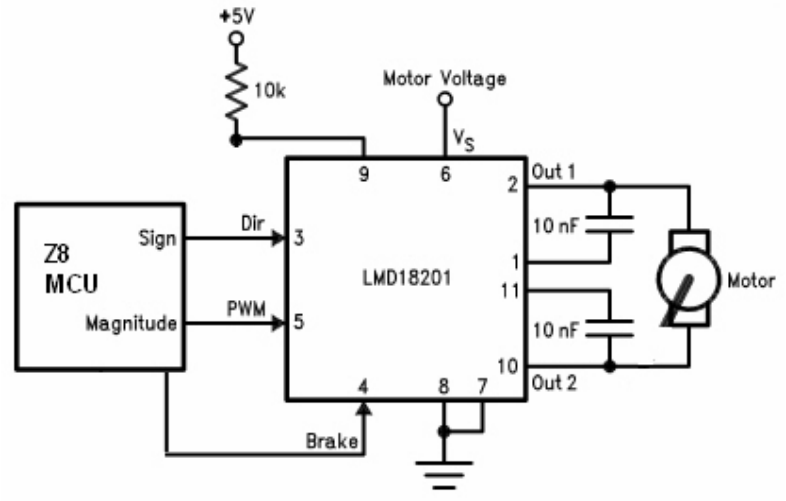


Fig 5.1 Basic motor control circuit using H-bridge

To test this circuit, at the first time, a signal generator was used as a PWM signal source, which provided the square wave signal with changeable duty. And, the Dir (Pin3) was connected to ground (GND) as "0" input. But, the motor was not stopped even when I set the duty of the square signal at 50%. More strangely, the H-bridge was consuming a large amount of current (more then 2A) and getting very hot in a short time. No reason was found after checking the whole circuit, and the only suspicion was it's a spoiled product. So, a new one was bought and has been tested working well.

5.2 Checking the ACE Output

Before transfer the data read from the ACE into the controller, it is necessary to check their accuracy. Here, Hyper Terminal®, which is a communication software integrated in Windows®, has been used to monitor these data.

On Z8 MCU development board, a serial port has been equipped to support the UART, and the connection is established between this serial port and the COM1 port in the host PC. Meanwhile, Hyper Terminal® is setup with the same baud rate as the UART in Z8 MCU, to display the data received from the microcontroller in arrays.

But, strangely, it seems the correlation between the digital output and the corresponding actual

position is random, for example, when the output is 1 (00000001), it means the actual position is 56; when the output is 127 (01111111), the actual position is 0. Therefore, a looking-up table has been created. (The looking-up table of the output codes to the absolute shaft position is listed in *Appendix B*.)

So, then, it's quite easy to understand the procedure. When the ACE shaft is rotating, the binary output data are referred to the Looking-Up table and converted into decimal data of the absolute position by Z8 MCU. And, those data displayed in Hyper Terminal® stand for the actual position of the shaft.

But due to the difficulties in compiling the C programs, this work has not been finished.

5.3 Pole and Cart System

Instead of controlling the pole and cart system directly by the ANFIS controller, it was decided to test it with manual control.

The motion control unit was connected as described in Chapter 5.1, and the direction was changed by switching Pin 3 of H-bridge from "0" (GND) to "1" (5V).

But, initially, the cart was not so swift and easy to control. The main reason was the weight. Improvement was made by cutting down four pieces of the cart to reduce its weight. (show in *Fig 5.2*)

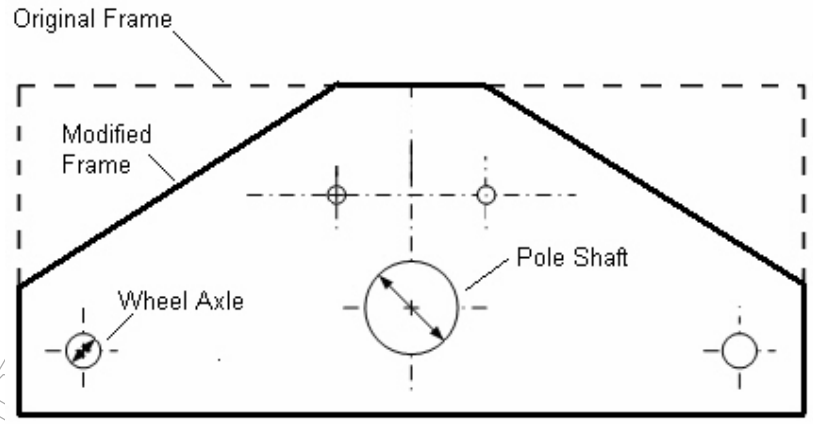


Fig 5.2 Modified Cart Frame

Another problem was the slippage between the driving wheel and the nylon string. Increasing the tensile force can decrease this slippage, but on the other hand, the motor power was not enough to deal with the heavy strain. So to maintain an amount of tensile force with certain flexibility, a spring has been added between one end of the string and the cart chassis. (*Fig 3.7*)

After all, the cart and pole system was tested, with manually governing the input signals. The

results are displayed below:

(PWM frequency is set at 100Hz, and Motor Voltage (V_S) is 12V)

PWM Duty Cycle (%)	Average Current (mA) (Output from H-Bridge)	Cart Speed (m/s) (Approximation)
50	0	0
75	362	0*
100	667	0.5

*: The expectative value here should be half of the maximum speed. But the friction is too much that overcomes the torque of the DC motor.

Thought the testing, it has demonstrated that changing the duty cycle of the PWM signal can change the speed of the DC motor. But the drawback is the friction, which prevents the cart speed to be linear with the PWM duty cycle, as expected in ideal status.

5.4 Chapter Summary

In this chapter, some parts of the whole hardware system have been tested. The problems and difficulties about the interface of ACE sensor have been presented and resolved. The cart has been modified to gain the enhancement in control ability. Thought the time is not sufficient to test the entire system, thought these tests, it can be said this inverted pendulum hardware part has been well built up and is ready to be implemented with the controller.

Chapter 6 PROJECT MANAGEMENT

6.1 Time Management

The original Gantt charts is shown below: (*Fig 5.1*)

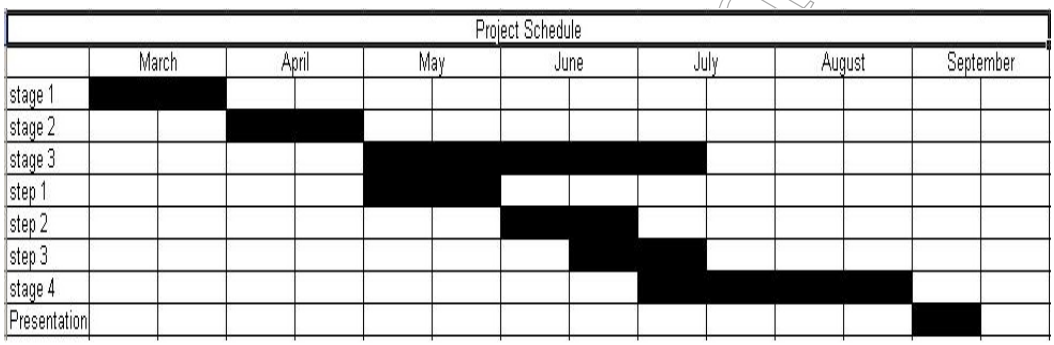


Fig 6.1 Original Gantt chart

- Stage 1: Studying theories and reading papers.
- Stage 2: Hardware design and selection.
- Stage 3: Testing the control methods.
  - Step 1: Training the neural network using the human help.
  - Step 2: Implementing the Self-learning technique in inverted pendulum system.
  - Step 3: Implementing the self-initialisation ability.
- Stage 4: Project report completion and further discussion.

But the actual time scale is shown below: (*Fig 6.2*)

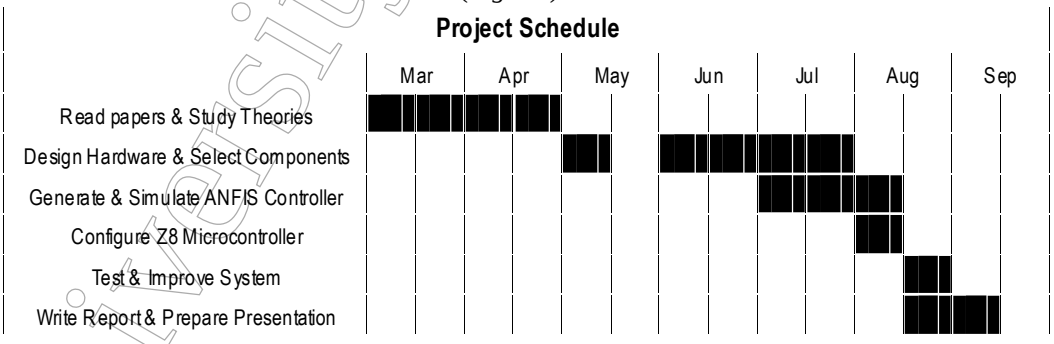


Fig 6.2 Updated Timescale

The time arrangement has been revised due to the following reasons:

1. Before starting with the practical work, a lot of readings are required in order to understand the basic concepts and fundamental theories. In April and May, without working in the hardware, I was struggling in how to implement the self-learning ability into the inverted pendulum system. But, unfortunately, even now, the complete training algorithm is not clearly understood; only a few of brief ideas are conceived, but have not been tested and verified.

2. During May, due to the final examinations, the progress was going slowly. From mid-June, I dedicated to this project and hurried up in the hardware design and components selection. But it took more then one month to finish the whole hardware system, a bit more then expectation. One reason was at the beginning, it was supposed to use Lego® to construct the hardware framework, but ten days had been spent before I gave up this attempt and made my own mechanical manufacture. Another reason was in assembly. Sometimes just a simple task like joining together two pieces of tiny things can take several hours! Also, orderings and the reordering (H-bridge) have postponed the schedule.

3. In parallel, the ANFIS controller was simulated in Matlab. It was not so hard in reading the documentations of Matlab® and Simulink® to find out the correct commends and operations. But, with the purpose to make the control system simpler, the ANFIS controller was tried again with only two inputs. This hypothesis was proven untenable later, but 10 days were consumed.

4. The biggest problem to accomplish this project is the coding and programming, which is significant in configuring the Z8 microcontroller. But, it's a shame that I have no knowledge in C/C^{++} language. Two weeks have been spent in reading the books on C, but obviously, C is not a two-week easy job, which needs fully comprehending and a lot of practicing. The best solution is finding some pieces of C-code from Internet. Even now, I am still trying my best to understand them and to make some modification according to my application.

6.2 Project Costing

The main components in the overall system and their costs are listed below: (*Table 5.1*)

Component	Manufacturer	Supply	Cost×Quantity
ZiLOG Z8 Encore® Development Kit (Z8F64200100KIT-C)	ZiLOG	UH Store	£34.99×1
Multi Ratio Motorgearbox	MFA	Maplin	£8.99×1
Absolute Contacting Encoder (ACE™)	BOURNS	RS	£6.10×1
Three Channel Optical Incremental Encoder Module	HP	Farnell	£15.47×1
Three Channel Codewheel	HP	Farnell	£12.31×1
LMD18201 Full Bridge	National Semiconductor	Farnell	£12.73×1
Bearing	RS	RS	£2.42×2
25mm Pulley	Rapid	Rapid	£1.95×1
Total Cost			£97.38

Table 5.1 Components and cost

Most of the components above are ordered from Project Lab C460 via Internet. Some other

elements are not included, such as the thermal-cut resistor for motor, 2 capacitors for H-bridge, data cables for communication, and the rails for cart, as well as the pendulum and the wheels from Lego® kit.

6.3 Equipments and Resources

Equipments and devices: (provided by laboratory)

- Two power supply, one for sensors and one for H-bridge
- A signal generator, for emulating PWM signal
- An oscillograph, for wave display
- A multimeter, for testing and examining
- Two bread-board and some wires, for circuit connection

Software: (Installed in a PC with Windows® XP OS)

- ZDS II - Z8Encore!® 4.7.0 for Z8 microcontroller configuration, and for debugging and compiling the C-programs
- Matlab® 6.5.1 & Simulink® for ANFIS controller simulation (Fuzzy Logic Toolbox included)
- Microsoft® Hyper Terminal for monitoring the input data from sensors
- Microsoft® Office: Word, PowerPoint, Project & Visio for report documentation

6.4 Chapter Summary

This chapter discusses the time management and explains the actual timescales, which has been revised according to the original one. The main components used in this project and their costs are listed. Also, the necessary devices and software resource are described.

Chapter 7 CONCLUSION AND FURTHER DISCUSSION

7.1 Conclusion

The aim of this project is to implement an ANFIS controller with self-learning ability into an inverted pendulum system, and this controller should be able to keep the system in dynamic balance, that means the pole is standing upright on the cart and the cart is moving around the center of the track. The principle of this control system is using a microcontroller (MC) to interface the real hardware parts with the software controller in a host PC. The status variables of the inverted pendulum system, which are obtained from the sensors, are sent to the ANFIS controller in MATLAB®, and simultaneously the output signal generated by the controller is sent back to the microcontroller, which immediately processes this signal into PWM signal for motion control. In this way, by handling the desired movement of the cart, the pendulum on it is kept upright in dynamic stable.

7.1.1 Achievement

Due to the practical problems, technical difficulties and time limit, this project is only partially completed. (Refer to Chapter 6.1 for detailed time management and updated timescale.) Even though, great efforts have been made to learn the new things and some outcomes have been achieved.

First of all, the frame of the whole hardware system has been built up. This work cost me much more time and energy then expected. To be honest, now I have realized that for any practical work, there are much more complication and difficulties then it looks like! In this project, all the necessary components are compared, studied, selected, purchased and assembled. The pole and cart part is designed and manufactured, and most of the requirements have been well satisfied, like the cart can move freely and swiftly along the rails, the pole can rotate in vertical with little friction, and the sensors are easy to be assembled. Also, all the circuits have been connected, and the motion control unit of H-bridge and DC motor has been assembled and tested, where the motor speed can vary smoothly according to the input PWM signal.

Furthermore, in software part, two types of ANFIS controller: one has 2-MF FIS; the other has 3-MF FIS, have been generated and evaluated in Matlab® environment. Both of them have satisfactory performance in controlling the simulated system model. The training data are collected from the “Cart & Pole” system, one of the demos in Simulink® Fuzzy Logic Toolbox, where the Fuzzy Logic Controller is being considered as an “expert”. On the other hand, the ANFIS controller with different patterns is tried and tested: using 4 inputs of cart and pendulum and using only the 2 inputs of the pendulum. It has been proved that in order to make all the control rules accurate, all the 4 inputs are necessary and both the states of the pole and cart have

to be considered in combination. Neglecting the cart position, only taking account the pendulum position, can never get a successful controller.

In addition, the research on the Z8 microcontroller is going deeper. Initially, it was just like a puzzle, but now, some basic theories have been studied and understood. The suitable ports and pins have been selected and assigned for each purpose; the control registers have been found and studied. So it has been known how to configure the Z8 MCU to perform the specialized tasks, like GPIO for receiving and transmitting data; UART for serial port communication; and Timer for PWM signal generation. Finally, the connection of the sensors, the Z8 MUC development board and the host PC has been established.

7.1.2 Overall Comment

Although the ANFIS controller has not been implemented into the real hardware system for testing, the simulation results of its controlling performance have confirmed that it is feasible to design and generate a successful controller for the inverted pendulum application using ANFIS training algorithm. Also, its self-learning ability has been demonstrated in the training procedure, where the ANFIS controller is trained automatically by only using the training data, without any human intervention or pre-knowledge. So, it can be concluded that for a non-linear system, which the control objective cannot be simply expressed as a set of functions defined over all states, ANFIS method is preferable and desirable in the controller design. With the benefit of its self-learning ability, if the appropriate training data is available, the ANFIS controller is easily to be achieved. Even there is no need to understand the inside control principles (the fuzzy rules). but the only problem, which is universal in any neural-network based structure and has a great effect on the controller's performance, is lying on how and where to gather and choose the optimal training data.

7.2 Further Discussion

7.2.1 Problems and Difficulties

For the hardware, in order to get ideal performance, some parts probably need a bit more consideration. The first thing is the DC motor. The present one is not so powerful to handle the cart, that even a moderate friction can stop it. (Refer to Chapter 5.3) And this motor goes quite hot after not long time working in full load. Another is the driving set. The slippery has not been completely eliminated even a spring is added, since it is not easy to find out the optimal tensile force that can provide enough strain but not too much.

To configure the Z8 microcontroller is another big issue. Some work has been carried out. But still some questions in how to build up the reliable communication from the Z8 MCU to the peripheral equipments are in doubt, such as how to configure the UART to transmit TWO signals to the serial port in alternate sequence; and how to convert the output signal *Force* from controller into PWM mode. Much more details have to be considered carefully and

comprehensively.

For any neural-network-based learning algorithm, the big problem is that the training outcome is always not absolutely predictable. (Discussed in chapter 2.4.2) The final controller was generated by 443 data sets, but it is still not sure whether these are the best training data, and whether the ANFIS controller can deal with all the possible situations, without any over-training or under-training phenomena.

Although the simulation in Matlab has been accomplished, there is still a big step to go before implementing the ANFIS controller into the real hardware system. It has to learn how the controller in Matlab/Simulink communicates with the PC serial port (COM1), and how to distribute the input data into each of the corresponding input port. Also, the output from the controller (*Force*) may need to be converted into a signal, which can be recognized and processed by the Z8 microcontroller, before being sent out.

7.2.2 Future Research and Improvement

The complete hardware system can be used as a platform for controllers' research and test. Different controllers, especially designed and generated in Matlab, can be easily implemented and tested, because the microcontroller works only as an interface rather then a standalone controller.

The biggest restriction to complete this project is the C language. Due to the deficiency of the essential knowledge of C language, the progress of writing programs for microcontroller has got stuck. Some efforts haven been made to read some books on C and try to understand them, but unfortunately, C language is really not an easy subject that can be learned in a short term. Many pieces of C code have been found form other sources, and modified according to this application. But a few errors arise in debugging and compiling, which haven't been resolved yet. So in the future, some more work is still required to carry on in this aspect, in order to make sure that all the programs are correct.

For the ANFIS controller design, the improvement can be done by modifying the FIS structure and optimizing the training data. Instead of the training data from the expert (like the demo in Matlab), further research is to find out how the ANFIS is capable to learn by its own, and which learning algorithm is the best one. Finally, even the self-initialization can be achieved. Some potential methods have been considered, which are presented in the literature on this subject, like reinforcement learning, self-adaptive learning, etc. But each of them needs to be tested and verified in the future.

Although the hardware system has been nearly completed, it still can be improved according to the problems described in Chapter 7.2.1. The research of DC motor is going on, and by calculating the maximum speed and torque of the cart and pole system, it has been decided the DC motor is at least 15V. The further experiments will focus on whether the cart speed is strictly linear to the duty ration of the PWM signal. About the driving unit problem, with the high-speed

DC motor, this simple nylon string is no longer sufficient for a more precise control; only the timer belt with gear pulleys is competent.

Reference

[1] John Nelson and L. Gordon Kraft, "Real-Time Control of an Inverted Pendulum System Using Complementary Neural Network and Optimal Techniques", *Proceedings of the American Control Conference*, Maryland, US, pp: 2553-2554, June, 1994.

[2] Wei Ji Chen, Lei Fang, Sek Un Cheyg, Kam Kin Lei, and Fei Zhou Zhang, "Personified Intelligent Control for an Inverted Pendulum System", *Proceedings of the 3rd World Congress on Intelligent Control and Automation*, Hefei, P.R. China, pp: 1702-1706, June 28-July 2, 2000.

[3] Jyh-Shing Roger Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference System", *IEEE Trans. on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665-685, May, 1993.

[4] Charles W. Anderson "Learning to Control an Inverted Pendulum Using Neural Networks", *Presented at the 1988 American Control Conference*, Atlanta, Georgia, June 15-17, 1988.

[5] Sigeru Omatu, Michifumi Yoshioka, "Stability of Inverted Pendulum by Neuro-PID Control with Genetic Algorithm", *IEEE: 0-7803-4859-1/98*, pp: 2142-2145, 1998

[6] Andrew James Appleby, "Dynamic Control of a "Helicopter"", *Final Year Report*, University of Herefordshire, April, 2004.

[7] Ankit Gorwadia, "Self Working Swing", *Final Year Report*, University of Herefordshire, April, 2005.

[8] Perminder Singh Thiara, "DSP Based Fuzzy Logic Controller for an Inverted Pendulum", *Final Year Report*, University of Herefordshire, April, 2001.

[9] ZiLOG Technical Note, "How to Use ZSL with the Z8 Encore!® MCU UART", *TN003702-0105*, www.ZiLOG.com

[10] ZiLOG Technical Note, "Using the GPIO Pins of the Z8 Encore!® MCU", *TN002401-0304*, www.ZiLOG.com

[11] ZiLOG Preliminary Product Specification, "Z8 Encore!® Z8F642x Series Microcontrollers with Flash Memory and 10-Bit A/D Converter", *PS019906-1003*, www.ZiLOG.com

[12] ZiLOG Application Note: "A DC Motor Controller Using A ZiLOG MCU", *AN006001-Z8X0400*, www.ZiLOG.com

[13] ZiLOG Application Note: "Using the Z8 Encore!® Timer", *AN013103-0104*, www.ZiLOG.com

[14] Agilent Technologies Inc. Technical Data: "Three Channel Optical Incremental Encoder Modules", www.semiconductor.agilent.com

[15] Data Sheet: "LMD18201 3A, 55V H-Bridge", www.mational.com

[16] Data Sheet: "Bourns Absolute Contacting Encoder ACE™", www.bourns.com

[17] http://www.mcmanis.com/chuck/robotics/tutorial/h-bridge/index.html, "H-Bridge Theory & Practice"

[18] MATLAB. "Fuzzy Logic toolbox For use with MATLAB (2000)", pp: 2-1 to 2-47

[19] The MathWorks Inc. www.mathworks.com

Bibliography

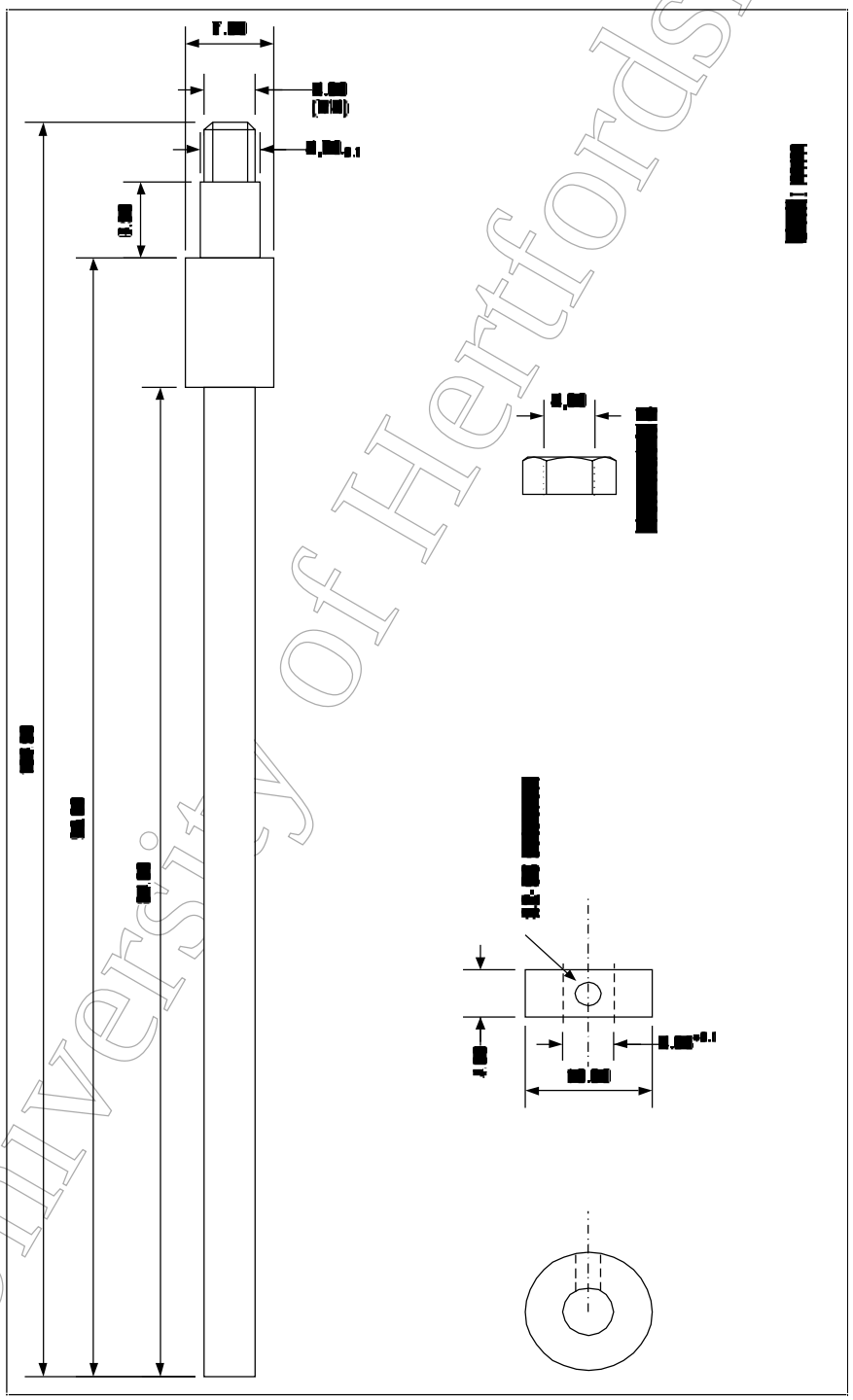
[1] Michael Negnevitsky, "Artificial Intelligence – A Guide to Intelligent Systems", First edition, Chapter 8, pp:275-284, 2002.

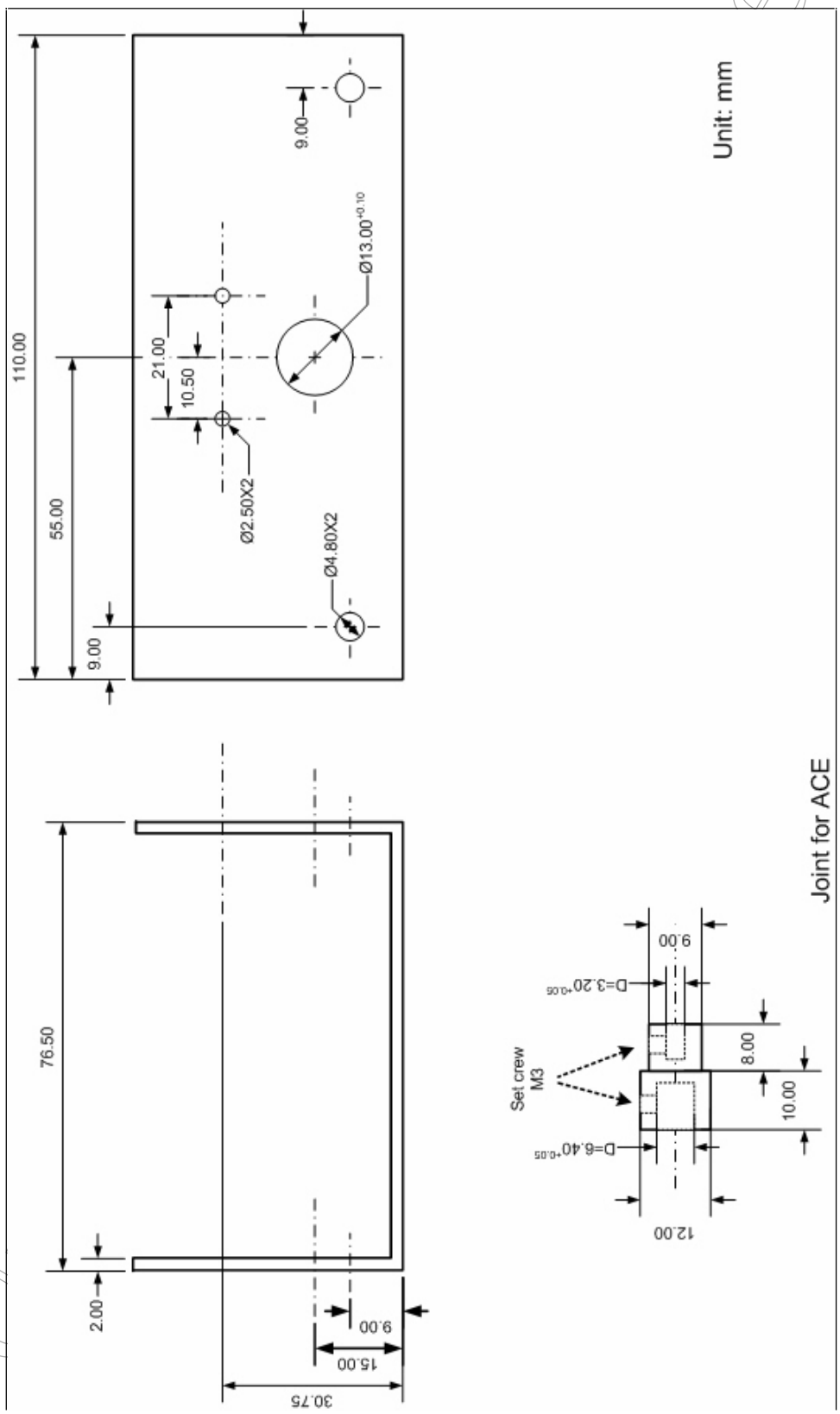
[2] Adrina Biran & Moshe Breiner, "Matlab 6 for Engineers", Third edition, chapter 14 & 15, pp: 547-597, 2002.

[3] Brian Overland, "C++ Without Fear: A Beginner's Guide That Makes You Feel Smart", First printing, Chapter 1 & 2, pp: 1-60, 2005.

Appendix

Appendix A: Mechanical Drawings




Unit: mm
Joint for ACE
110.00
55.00
9.00
9.00
21.00
10.50
Ø13.00+0.10
Ø2.50X2
Ø4.80X2
76.50
2.00
9.00
15.00
30.75
9.00
D=3.20+0.05
8.00
10.00
Set crew
M3
D=6.40+0.05
12.00

Appendix B: Looking-Up Table of ACE

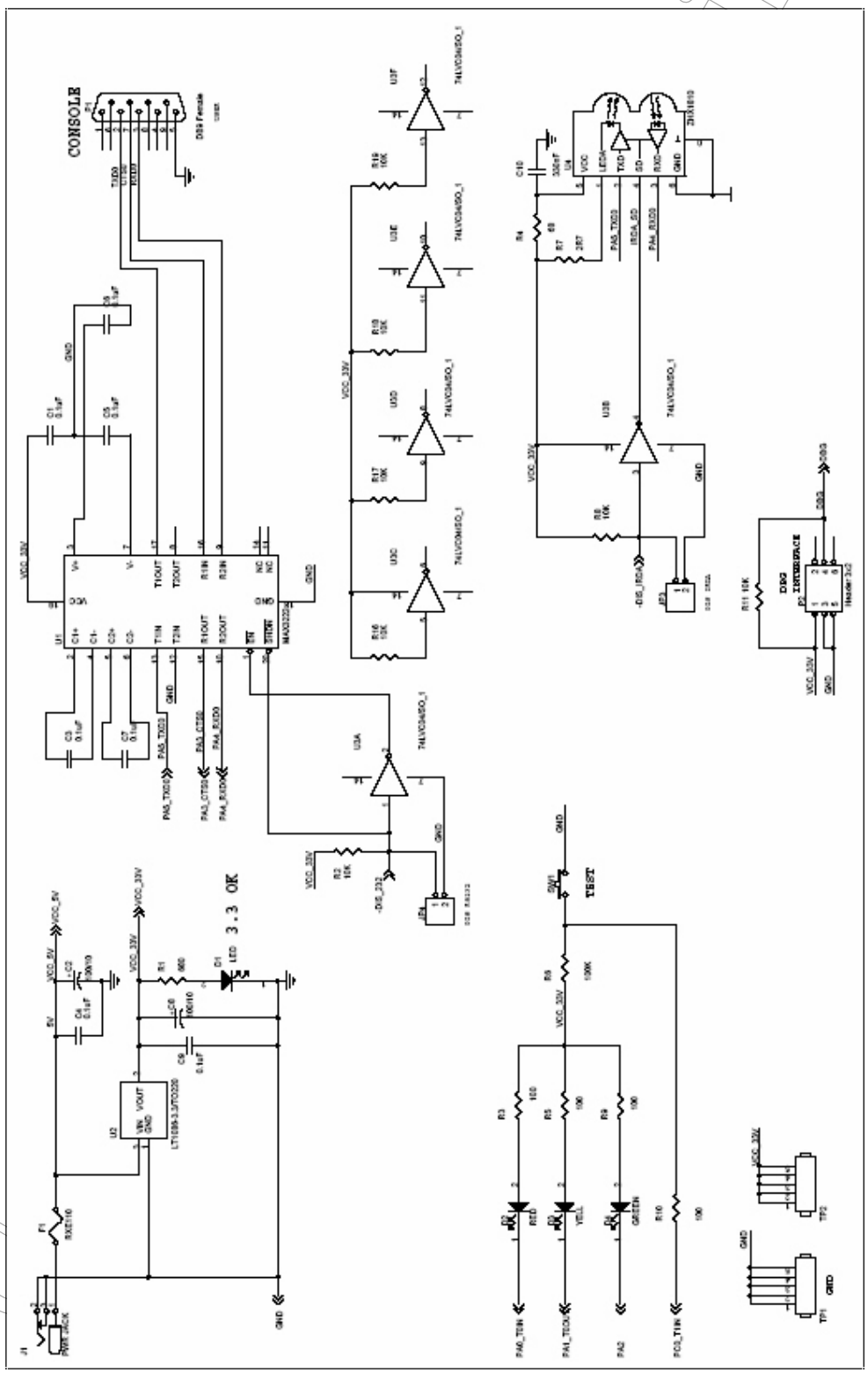
Pin Output Code For ACE-128

Bit/Pin correlation: b7 b6 b5 b4 b3 b2 b1 b0 = p8 p7 p6 p5 p4 p3 p2 p1
A binary "1" denotes an "open" switch and a binary "0" denotes a "closed" switch.
Positions 0-127 are seen by a clockwise rotation of the shaft.

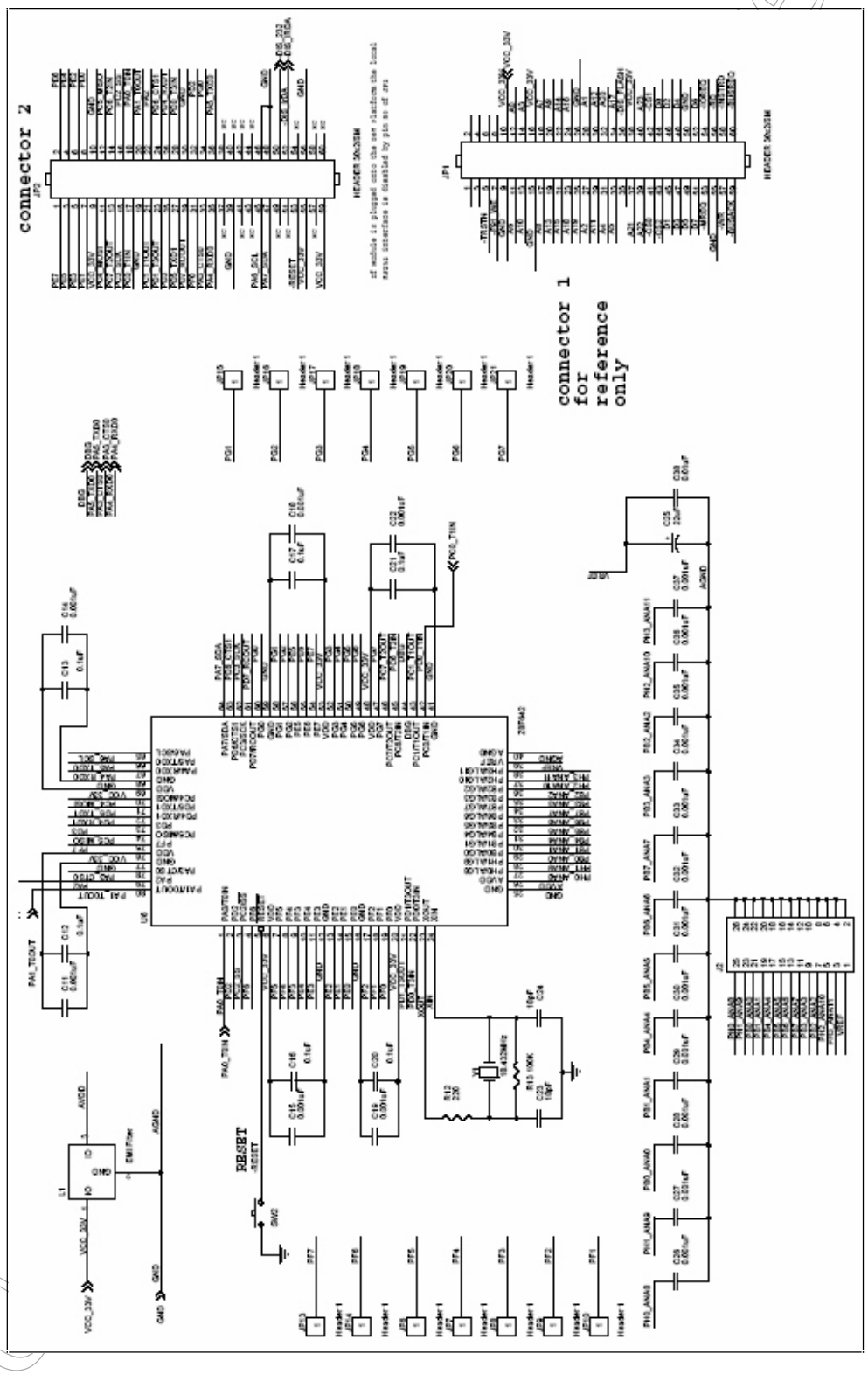
Position	p8	p7	p6	p5	p4	p3	p2	p1	Decimal Output
0	0	1	1	1	1	1	1	1	127
1	0	0	1	1	1	1	1	1	63
2	0	0	1	1	1	1	1	0	62
3	0	0	1	1	1	0	1	0	58
4	0	0	1	1	1	0	0	0	56
5	1	0	1	1	1	0	0	0	184
6	1	0	0	1	1	0	0	0	152
7	0	0	0	1	1	0	0	0	24
8	0	0	0	0	1	0	0	0	8
9	0	1	0	0	1	0	0	0	72
10	0	1	0	0	1	0	0	1	73
11	0	1	0	0	1	1	0	1	77
12	0	1	0	0	1	1	1	1	79
13	0	0	0	0	1	1	1	1	15
14	0	0	1	0	1	1	1	1	47
15	1	0	1	0	1	1	1	1	175
16	1	0	1	1	1	1	1	1	191
17	1	0	0	1	1	1	1	1	159
18	0	0	0	1	1	1	1	1	31
19	0	0	0	1	1	1	0	1	29
20	0	0	0	1	1	1	0	0	28
21	0	1	0	1	1	1	0	0	92
22	0	1	0	0	1	1	0	0	76
23	0	0	0	0	1	1	0	0	12
24	0	0	0	0	0	1	0	0	4
25	0	0	1	0	0	1	0	0	36
26	1	0	1	0	0	1	0	0	164
27	1	0	1	0	0	1	1	0	166
28	1	0	1	0	0	1	1	1	167
29	1	0	0	0	0	1	1	1	135
30	1	0	0	1	0	1	1	1	151
31	1	1	0	1	0	1	1	1	215
32	1	1	0	1	1	1	1	1	223
33	1	1	0	0	1	1	1	1	207
34	1	0	0	0	1	1	1	1	143
35	1	0	0	0	1	1	1	0	142
36	0	0	0	0	1	1	1	0	14
37	0	0	1	0	1	1	1	0	46
38	0	0	1	0	0	1	1	0	38
39	0	0	0	0	0	1	1	0	6
40	0	0	0	0	0	0	1	0	2
41	0	0	0	1	0	0	1	0	18
42	0	1	0	1	0	0	1	0	82
43	0	1	0	1	0	0	1	1	83
44	1	1	0	1	0	0	1	1	211
45	1	1	0	0	0	0	1	1	195
46	1	1	0	0	1	0	1	1	203
47	1	1	1	0	1	0	1	1	235
48	1	1	1	0	1	1	1	1	239
49	1	1	1	0	0	1	1	1	231
50	1	1	0	0	0	1	1	1	199
51	0	1	0	0	0	1	1	1	71
52	0	0	0	0	0	1	1	1	7
53	0	0	0	1	0	1	1	1	23
54	0	0	0	1	0	0	1	1	19
55	0	0	0	0	0	0	1	1	3
56	0	0	0	0	0	0	0	1	1
57	0	0	0	0	1	0	0	1	9
58	0	0	1	0	1	0	0	1	41
59	1	0	1	0	1	0	0	1	169
60	1	1	1	0	1	0	0	1	233
61	1	1	1	0	0	0	0	1	225
62	1	1	1	0	0	1	0	1	229
63	1	1	1	1	0	1	0	1	245

Position	p8	p7	p6	p5	p4	p3	p2	p1	Decimal Output
64	1	1	1	1	0	1	1	1	247
65	1	1	1	1	0	0	1	1	243
66	1	1	1	0	0	0	1	1	227
67	1	0	1	0	0	0	1	1	163
68	1	0	0	0	0	0	1	1	131
69	1	0	0	0	0	1	0	1	139
70	1	0	0	0	1	0	0	1	137
71	1	0	0	0	0	0	0	1	129
72	1	0	0	0	0	0	0	0	128
73	1	0	0	0	0	0	1	0	132
74	1	0	0	0	1	0	1	0	148
75	1	1	0	0	1	0	1	0	212
76	1	1	1	0	1	0	1	0	244
77	1	1	1	1	0	0	0	0	240
78	1	1	1	1	0	0	1	0	242
79	1	1	1	1	1	0	1	0	250
80	1	1	1	1	1	0	1	1	251
81	1	1	1	1	1	0	0	1	249
82	1	1	1	1	0	0	0	1	241
83	1	1	0	1	0	0	0	1	209
84	1	1	0	0	0	0	0	1	193
85	1	1	0	0	0	1	0	1	197
86	1	1	0	0	0	1	0	0	196
87	1	1	0	0	0	0	0	0	192
88	0	1	0	0	0	0	0	0	64
89	0	1	0	0	0	0	1	0	66
90	0	1	0	0	1	0	1	0	74
91	0	1	1	0	1	0	1	0	106
92	0	1	1	1	1	0	1	0	122
93	0	1	1	1	1	0	0	0	120
94	0	1	1	1	1	0	0	1	121
95	0	1	1	1	1	1	0	1	125
96	1	1	1	1	1	1	0	1	253
97	1	1	1	1	1	1	0	0	252
98	1	1	1	1	1	0	0	0	248
99	1	1	1	0	1	0	0	0	232
100	1	1	1	0	0	0	0	0	224
101	1	1	1	0	0	0	1	0	226
102	0	1	1	0	0	0	1	0	98
103	0	1	1	0	0	0	0	0	96
104	0	0	1	0	0	0	0	0	32
105	0	0	1	0	0	0	0	1	33
106	0	0	1	0	0	1	0	1	37
107	0	0	1	1	0	1	0	1	53
108	0	0	1	1	1	1	0	1	61
109	0	0	1	1	1	1	0	0	60
110	1	0	1	1	1	1	0	0	188
111	1	0	1	1	1	1	1	0	190
112	1	1	1	1	1	1	1	0	254
113	0	1	1	1	1	1	1	0	126
114	0	1	1	1	1	1	0	0	124
115	0	1	1	1	0	1	0	0	116
116	0	1	1	1	0	0	0	0	112
117	0	1	1	1	0	0	0	1	113
118	0	0	1	1	0	0	0	1	49
119	0	0	1	1	0	0	0	0	48
120	0	0	0	1	0	0	0	0	16
121	1	0	0	1	0	0	0	0	144
122	1	0	0	1	0	0	1	0	146
123	1	0	0	1	1	0	1	0	154
124	1	0	0	1	1	1	1	0	158
125	0	0	0	1	1	1	1	0	30
126	0	1	0	1	1	1	1	0	94
127	0	1	0	1	1	1	1	1	95

Appendix C: Schematic of Z8 MCU Development Board



Cont.


connector 2
connector 1
for
reference
only
RESET

Appendix D: Flowchart to Configure Timers