การออกแบบตัวควบคุมข้อต่อแขนหุ่นยนต์ MITSUBISHI PA10

นายสีเรระทา พัล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2558

Design of joint controller for Mitsubishi PA10 robot arm

Mr. Sereiratha Phal

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2015

Thesis Title	Design of joint controller for Mitsubishi PA10 robot arm
By	Mr. Sereiratha Phal
Field of Study	Electrical Engineering
Thesis Advisor	Assistant Professor Manop Wongsaisuwan, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

.....Dean of the Faculty of Engineering
(Associate Professor Supot Teachavorasinskun, D.Eng.)

THESIS COMMITTEE

.....Chairman
(Professor David Banjerdpongchai, Ph.D.)

.....Thesis Advisor
(Assistant Professor Manop Wongsaisuwan, Ph.D.)

.....External Examiner
(Sirichai Pornsarayouth, Ph.D.)

สีเรระทา พัล : การออกแบบตัวควบคุมข้อต่อแขนหุ่นยนต์ MITSUBISHI PA10 (Design of joint controller for Mitsubishi PA10 robot arm) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ผศ. ดร. มานพ วงศ์สายสุวรรณ, 63 หน้า.

ในวิทยานิพนธ์นี้ เรานำเสนอการออกแบบวงจรและอัลกอริธึมสำหรับการควบคุมข้อต่อของแขนกลรุ่น MISUBISHI PA10 ซึ่งข้อต่อทั้งหมดของแขนกลรุ่นนี้ถูกสวมเข้ากับมอเตอร์ซิงโครนัสชนิดแม่เหล็กถาวร (PMSM) สำหรับการหมุนและภายในมอเตอร์แต่ละตัวประกอบไปด้วยสองรีโซล์เวอร์ (Resolver) เพื่อใช้สำหรับอุปกรณ์ป้อนกลับ โดยที่รีโซล์เวอร์ตัวแรกใช้วัดมุมของข้อต่อแขนกลและรีโซล์เวอร์ตัวที่สองทำหน้าวัดมุมของโรเตอร์ของมอเตอร์ที่ข้อต่อนั้น ๆ ในการออกแบบตัวควบคุม ลำดับแรกเราสนใจเกี่ยวกับหลักทฤษฎีและอุปกรณ์ที่จำเป็นสำหรับการขับเคลื่อนข้อต่อของแขนกลเพื่อที่จะสร้างวงจรควบคุมสำหรับตัวควบคุมได้ ลำดับถัดมาคือการเขียนโปรแกรมลงบนวงจรควบคุมเพื่อควบคุมข้อต่อของแขนกลให้ปฏิบัติตามคำสั่งที่ถูกส่งมาจากคอมพิวเตอร์ผ่านช่องทางการสื่อสารแบบ Serial ตัวควบคุมข้อต่อแขนกลมีความสำคัญอย่างมากในฐานะที่เป็นตัวควบคุมย่อยจากตัวควบคุมทั้งหมดที่คอยควบคุมแขนกลทั้งแขน เป้าหมายในอนาคตของการออกแบบตัวควบคุมข้อต่อแขนกลในวิทยานิพนธ์คือเพื่อให้สามารถนำมาใช้ขับเคลื่อนแขนกล PA10 ทั้งแขนได้เมื่อนำมาใช้ร่วมกับตัวควบคุมข้อต่ออื่น ๆ ในการควบคุมข้อต่อแขนกลที่เหลืออีกรวมเจ็ดข้อต่อ

ภาควิชา วิศวกรรมไฟฟ้า ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมไฟฟ้า ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

ปีการศึกษา 2558

5670557721 : MAJOR ELECTRICAL ENGINEERING
KEYWORDS: PERMANENT MAGNET SYNCHRONOUS MOTOR / PULSE-WIDTH MODULATION / , SINE PULSE-WIDTH MODULATION / MICRO CONTROLLER UNIT

SEREIRATHA PHAL: Design of joint controller for Mitsubishi PA10 robot arm. ADVISOR: ASST. PROF. MANOP WONGSAISUWAN, Ph.D., 63 pp.

In this thesis we present the design of circuit and algorithms for controlling the joint of Mitsubishi PA10 Robot Arm. To control the entire Mitsubishi PA10 arm robot, which all the joints equip with a permanent magnet synchronous motor (PMSM) for rotation and two resolvers as feedback devices, one for joint angle position of the joint and another one for rotor angular position of the motor at the jont. For the controller design, the first focus is on theory and hardware needed for driving the robot joint in order to build the circuit board of the controller. The second part is to program the board for controlling the joint of the robot according to command sent from the PC via serial communication. The joint controller is really important as it is the sub-controller for the whole-robot controller system of the robot. The future aim of the joint controller design in this thesis is to be able to drive the whole PA10 robot arm when combining seven of this controller to meet the seven joints of the robot.

Department: Electrical Engineering Student's Signature .....
Field of Study: Electrical Engineering Advisor's Signature .....
Academic Year: 2015

ACKNOWLEDGEMENTS

Most of all, I am grateful to my principle advisor, Asst. Prof. Dr. Manop Wongsaisuwan for his help and encouragement in every technical and academic problem I faced over the past two years. He always made his time available to discuss my research working with him has been exhilarating experience. I feel lucky to have had my path cross him.

I would like to express my sincere thank to Dr.Sirichai Pornsarayouth for being and always helpful along the way of my research. His inputs, encouragement in technical, and helpful suggestions are really important on my research.

I would like to thank Prof. Dr. David Banjerdpongchai and Dr. Pasu Boonvisut for being on a committee for my oral exam. Their inputs, helpful suggestions, and comments on the proposal exam are invaluable for the following steps of my research.

Finally, especial credit goes to my family: my parents, my brothers, and sisters for their support throughout the production of this dissertation and encouragement for my study in Thailand.

This research was carried out at CSRL, the Control System Research Laboratory at Chulalongkorn University and was fully funded through AUN/SEED-Net scholarship, which is gratefully acknowledged.

CONTENTS

Page

THAI ABSTRACT iv

ENGLISH ABSTRACT v

ACKNOWLEDGEMENTS vi

CONTENTS vii

List of Figures ix

List of Tables xi

CHAPTER 1 INTRODUCTION 1

1.1 Motivation 1

1.2 Background and Literature Review 2

1.3 Thesis Objectives 5

1.4 Scope of Thesis 5

1.5 Methodology 5

1.6 Expected Outcome 5

1.7 Organization of Dissertation 5

CHAPTER 2 THEORIES AND HARDWARE DESIGN 7

2.1 PMSM 7

2.2 Inverter Driver 8

2.3 RS485 11

2.4 Resolver 13

2.5 Microcontroller-Resolver Interface 15

2.6 Power Supply of the Controller Board 18

2.7 PCB Design 20

2.7.1 KiCad 20

2.7.2 PCB Design Workflow 22

CHAPTER 3 ALGORITHM AND SOFTWARE DESIGN 24

3.1 Atmel Studio 24

3.2 Acquire Position and Velocity Output of Resolver from AD2S1205 25

3.3 Sin Wave Generating Technique 29

Page

3.4 Driving Joint Motor 32
3.4.1 Theory for Driving 32
3.4.2 Experiment and Design 33
3.5 Joint Controller Workflow 36
3.6 Command from Matlab 40
CHAPTER 4 CONTROLLER DESIGNING RESULT AND DISCUSSION 41
4.1 Controller Board 41
4.2 Upload Software to the Board 43
4.2.1 SAM-BA Mandatory Software Setup 43
4.2.2 USB Driver Checking 43
4.2.3 Software Upload 44
4.3 Expense Cost 47
4.4 Experiment and Result Discussion 48
CHAPTER 5 CONCLUSIONS 51
REFERENCES 52
APPENDIX 53
List of Publication 54
Controller Board in 3D 55
Controller Schematic 56
Component List 61
VITA 63

List of Figures

Figure 1: Mitsubishi PA10 Arm Robot 2
Figure 2: PA10 joint configuration 4
Figure 3: Physical of PMSM 7
Figure 4: Sine Pulse Width Modulation (SPWM) 8
Figure 5: Three-phase voltage source PWM Inverter 8
Figure 6: Switching schematic design 9
Figure 7: Inverter Driver Board by HiveGround 10
Figure 8: RS485 network topology 11
Figure 9: MAX3485 pin configuration 12
Figure 10: Connection between MAX3485 and MCU 12
Figure 11: Classical Resolver and Variable Reluctance Resolver 13
Figure 12: Resolver signals 14
Figure 13: Resolver interface system block diagram 15
Figure 14: AD2S1205 Interface and Buffer Circuit Based on AD8397 op amp 16
Figure 15: Power Supply Structure 18
Figure 16: Linear Regulator LM1117MPX50 for 5V Supply 19
Figure 17: Linear Regulator LD1117AS33 for 3.3V Supply 19
Figure 18: Linear Regulator BU12TD3WGTR for 1.2V Supply 19
Figure 19: Kicad 4.0.1 20
Figure 20: Electronic Schematic Editor 21
Figure 21: Printed Circuit Editor 21
Figure 22: Workflow Design of Making PCB 22
Figure 23: Bill of Materials 23
Figure 24: Atmel Studio 6 24
Figure 25: Serial Port Read Timing of AD2S1205 25
Figure 26: Data Reading Flowchart 27
Figure 27: Computation time for Sin(x) using Taylor Series 30
Figure 28: Sin(x) PWM 30

Figure 29: Total time using in driving the joint 31
Figure 30: Rotor and Stator Flux Vectors 32
Figure 31: Motor Torque vs Angle between Rotor and Stator Flux Vectors 32
Figure 32: Experiment for finding rotor initial point 34
Figure 33: Joint Motor Driving Flowchart 36
Figure 34: Control System of the Controller 37
Figure 35: Controller Structure 37
Figure 36: Matlab GUI for Sending Command 40
Figure 37: Joint Controller Board 41
Figure 38: SAM-BA 2.11 43
Figure 39: Summary of USB driver checking 43
Figure 40: Verify USB driver in Computer Management 44
Figure 41: Upload Software Process 45
Figure 42: Connect the Controller board in SAM-BA 46
Figure 43: Upload .bin File 46
Figure 44: PWM Input Signal When Moving Forward 48
Figure 45: Joint Position When Moving Forward 48
Figure 46: Motor Velocity When Moving Forward 49
Figure 47: PWM Input Signal When Moving Backward 49
Figure 48: Joint Position When Moving Backward 50
Figure 49: Motor Velocity When Moving Backward 50

List of Tables

Table 1: Specifications of Mitsubishi PA10 Robot Arm 3
Table 2: Resolver Key Parameters 15
Table 3: Timing Diagram of AD2S1205 26
Table 4: Excitation Frequency Selection 42
Table 5: Expanse for Controller Board Production 47

CHAPTER 1
INTRODUCTION

1.1 Motivation

Robots play extremely important role in the manufacturing process due to the increased demand of high quantity and quality of goods from consumers. If the workers in the production process work alone, the products from the process cannot produce fast enough to meet the demand. To make the production process produces quantities of product faster, the robot is a great alternative solution for this problem as it can work faster and continuously longer than human.

In order to be able to use the robot to operate in the process, the robot needs to have robot controller. Normally the price of the controller is quite high and in most cases the manufacturers of the robot do not provide technical details document of the equipment inside. So it is difficult to repair or maintain when there is problem.

In CSRL lab there is a Mitsubishi PA10-7C robot arm, which is a 7 degree of freedom (DoFs) articulate robot arm, made by Mitsubishi Heavy Industries. The robot consists 7 revolute joints. Each joint of the robot consists of one Permanent Magnet Synchronous Motor (PMSM), two resolvers as position and velocity feedback, one is attached to the rotor of the motor and the other one is attached to the motor gear, and one electrical brake [1]. This robot in the lab does not function as there is no controller to make this robot work.

The robot without controller is useless because it cannot perform any task and even need some space to store it in the lab. The robot in the lab should function and customizable in order for using it in educational purpose, but making this robot work again is really challenging as there is no information of how the controller of the robot is made from the manufacturer, so the hardware and software must be designed again according to the mechanical and configuration of parts of the robot.

Thus the study of the mechanical part configuration of the robot, designing new electronic board (Hardware), and programming for controlling method for the robot (Software) is needed.

1.2 Background and Literature Review

Mitsubishi PA10

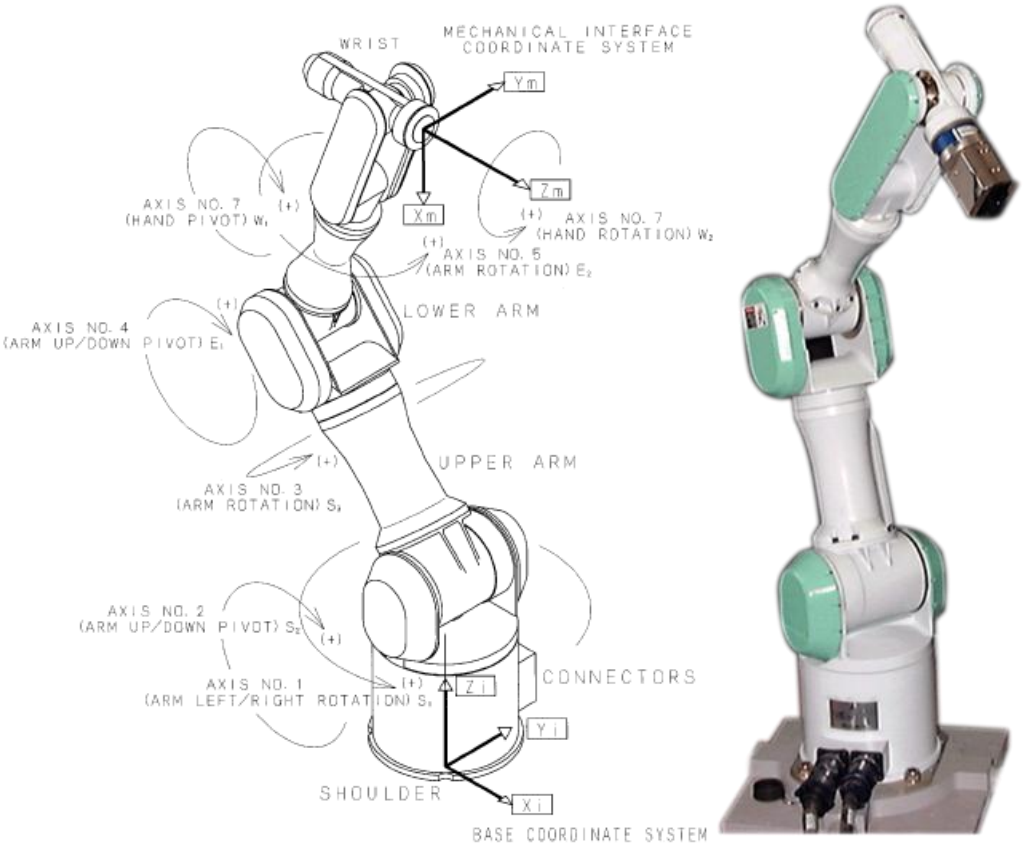


Figure 1: Mitsubishi PA10 Arm Robot [1, 2]

Mitsubishi PA10 is as dexterous as human hand even under adverse environment. With the arm length of 950mm the same as human hand, the 7-joint specification allows the work to be performed while avoiding obstacles. The machine (arm only) can be operated even under environment requiring dust-proof/drip proof (standard specification), explosion proof/ water proof/ cleanness (optional) [1].

The deal weight is only 35Kg which is light weight and potable. It can lift the weight up to 10Kg. Figure *1* shows the Mitsubishi PA10 Arm robot (right) and the joint configuration of the robot (left) and Table 1 presents the specification of the robot arm.

Table 1: Specifications of Mitsubishi PA10 Robot Arm

<table>
<tr><th>Item</th><th>Specifications</th></tr>
<tr><td>Model</td><td>Mitsubishi PA-10A-ARM</td></tr>
<tr><td>Type</td><td>Vertical multi-joint type</td></tr>
<tr><td>Configuration</td><td>Dust-proof/ drip-proof construction (explosion-proof/ water proof as optional)</td></tr>
<tr><td>Number of joint</td><td>7</td></tr>
<tr><td>Joint configuration</td><td>R-P-R-P-R-P-R from robot mounting side (R: rotation, P: Pivot)</td></tr>
<tr><td>Joint names</td><td>S1-S2-S3-E1-E2-W1-W2 from robot mounting-side (S: shoulder joint, E: elbow joint, W: wrist joint)</td></tr>
<tr><td>Arm length</td><td>Shoulder reach: 315mm (base surface to S2)
Upper arm : 450mm (S2 to E1 axes)
Lower arm : 500mm (E1 to W1 axes)
Wrist reach : 80mm (W1 to mechanical interface side)</td></tr>
<tr><td>Joint operating range and max operation speed</td><td>
<table>
<tr><th rowspan="2">Axis name</th><th colspan="3">Limit (angle)</th><th rowspan="2">Max operating speed (rad/sec)</th></tr>
<tr><th>Mechanical limit</th><th>Servo limit</th><th>Software limit</th></tr>
<tr><td>S1 (rotation)</td><td>+-180</td><td>+-178</td><td>+-177</td><td>+-1</td></tr>
<tr><td>S2 (swing)</td><td>+-90</td><td>+-92</td><td>+-91</td><td>+-1</td></tr>
<tr><td>S3 (rotation)</td><td>+-180</td><td>+-178</td><td>+-174</td><td>+-2</td></tr>
<tr><td>E1 (swing)</td><td>+-143</td><td>+-138</td><td>+-137</td><td>+-2</td></tr>
<tr><td>E2 (rotation)</td><td>+-270</td><td>+-256</td><td>+-255</td><td>+-2л</td></tr>
<tr><td>W1 (swing)</td><td>+-180</td><td>+-166</td><td>+-165</td><td>+-2 л</td></tr>
<tr><td>W2 (rotation)</td><td>+-limitless rotation</td><td>+-360</td><td>+-360</td><td>+-2 л</td></tr>
</table>
</td></tr>
<tr><td>Max. integrated speed</td><td>1550mm/s</td></tr>
<tr><td>Load capacity</td><td>10Kg</td></tr>
<tr><td>Drive method</td><td>AC servo motor with non-excitation break/ brushless resolver</td></tr>
<tr><td>Sensor</td><td>Output axis brushless resolver</td></tr>
<tr><td>Ambient temp.</td><td>0 to 5C^{o}</td></tr>
<tr><td>Humidity</td><td>30 to 90% RH (no condensation)</td></tr>
<tr><td>Storage temp</td><td>-10 to 60 C^{o}</td></tr>
<tr><td>Mass of main body</td><td>36Kg</td></tr>
</table>

External appearance	Finish by aluminum alloy painting

Robot Joint Configuration

Each joint of the robot consists of 5 main important parts (Figure 2):

- 1 Harmonic Drive or Harmonic Gear: with the reduction ratio is given by

$$Reduction\ ratio = \frac{Flex\ spline\ teeth - Circular\ spline\ teeth}{Flex\ spline\ teeth} \quad (1)$$

- 1 PMSM
- 2 Resolvers (one for joint angle feedback and another one is for motor angle feedback)
- 1 Brake
- Limit switch (for S1, S3, and E2)

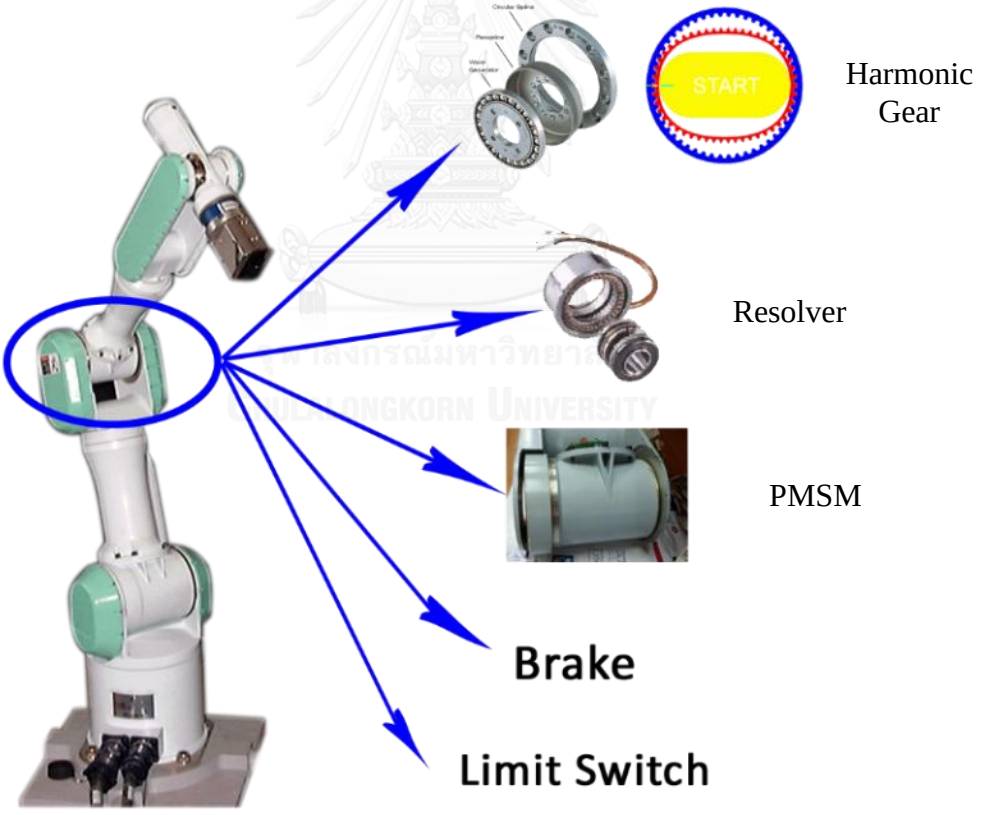


Figure 2: PA10 joint configuration

1.3 Thesis Objectives

The objective of this thesis is to study the characteristic of all joints of PA10 Robot Arm in order to construct a low cost joint robot controller prototype based on simple controller for the robot arm which can drive the joint of the robot according to giving commend from the PC to the specific ordered position. This controller will bring the old robot to work again and can be further developed to control the whole robot by combining the number of controller needed for all joint of the robot. The controller will be programmable for the user, so it can be used in the lab for education purpose as well.

1.4 Scope of Thesis

1. Construct a robot joint controller prototype which is able to drive and control the joint of Mitsubishi PA10 Robot Arm
2. The cost of the controller is no more than 10,000 baht.
3. Develop a simple programing code in MatLab for communication interface between PC and controller in order the controller to get command from PC.

1.5 Methodology

1. Literature review on mechanical design and configuration of Mitsubishi PA10 Robot Arm
2. Literature review on circuit design needed for controller
3. Build a prototype of joint controller for the robot
4. Program the board according to suitable algorithm for position control of the robot joint.

1.6 Expected Outcome

1. Prototype of a joint controller for the of Mitsubishi PA10 Robot Arm
2. Plug and play circuit for robot joint control

1.7 Organization of Dissertation

In this section, it presents the rest of this thesis arrangement. Chapter 2 is focusing on theories and design of hardware for the main component and electronic needed for the project such as PMSM, Resolves, RS485, Resolver-MCU interface, and PCB

designing process. The detail of algorithm and software design for driving motor with maximum torque, calculation technique for microcontroller and controlling flow of the robot joint are discussed in Chapter 3. Chapter 4 talks about the results of the hardware design (the prototype of the joint controller board), result from the experiments and discussion. The conclusion of this dissertation, as well as the future work, are in Chapter 5.

CHAPTER 2
THEORIES AND HARDWARE DESIGN

2.1 PMSM

Permanent Magnet Synchronous Motor (PMSM) is used at the joint of the robot moving the joint. PMSM is brushless motor which consist of a permanent magnet on the rotor and coil windings on the stator as shows in Figure 3. It requires alternating stator currents in order to produce constant torque.

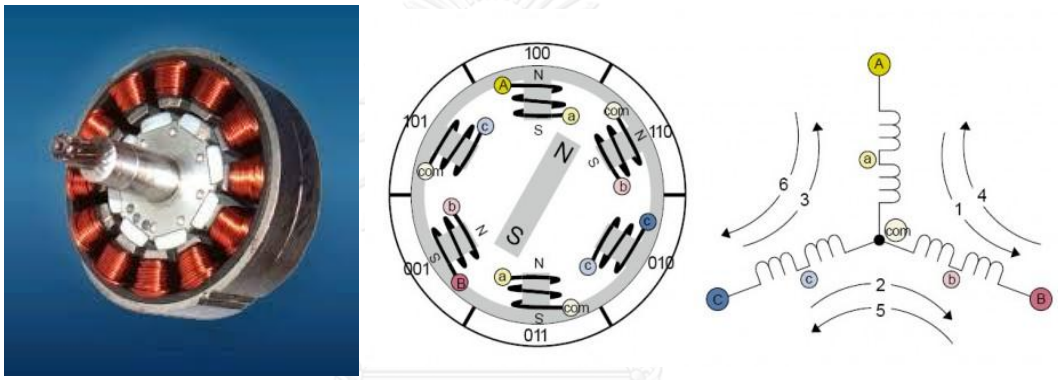


Figure 3: Physical of PMSM [3]

There are some characteristics of the PMSM such as:

- Fed with sinusoidal current
- Continuous stator flux at angle change
- All phases of the PMSM are on at the same time
- Can achieve low toque ripple
- High achievable speed
- Low noise when operating
- Good for position servo
- Need high resolution feed back

Because all these characteristics make PMSM suitable for many kind of robot application especially for position servo application [4].

2.2 Inverter Driver

As it is already said above section the nature of the PMSM is that it needs to be fed by the sinusoidal current according to the position of the rotor. That is why it needs the three-phase voltage source PWM inverter to switch DC sources on and off to make PWM signal with the form of sinusoidal, called Sine Pulse-Width Modulation (SPWM).The SPWM signal is shown in the Figure 4. The circuit model of a typical three-phase voltage source PWM inverter as shown in Figure 5, consists of 6 power switches that shape the output, which are controlled by the switch variable a, a', b, b', c, and c'. When an upper transistor is switched on (a, b, c, is 1) the corresponding lower switches are turned off (a', b', c' is 0).

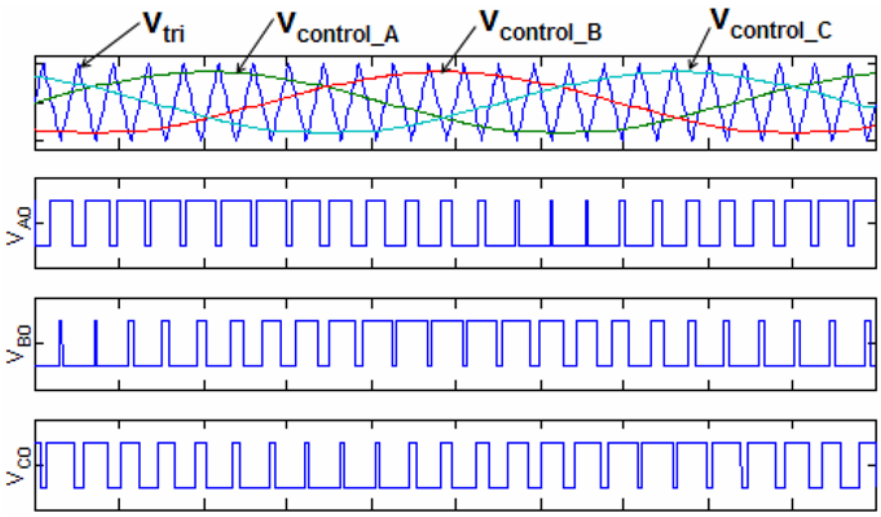


Figure 4: Sine Pulse Width Modulation (SPWM) [5]

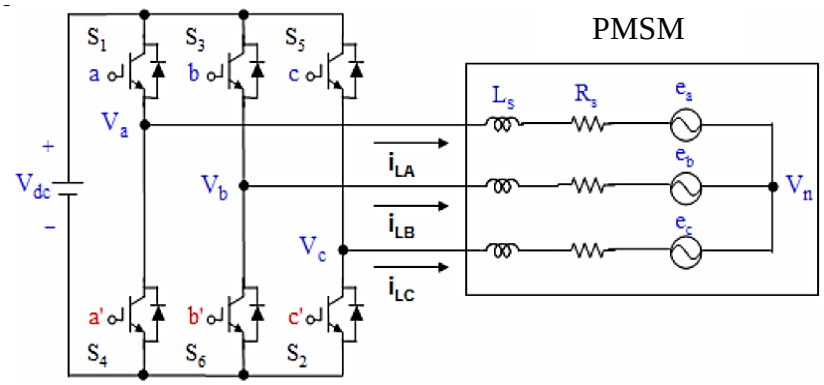


Figure 5: Three-phase voltage source PWM Inverter

Figure 5 presents the three phase voltage source PWM inverter which consists of 6 switches ,but in order to be able to drive the switch from MCU each switch need more components and complexity. The switch must be able to turn on or turn off by standard logic which is controlled by MCU, the N type of power Field-Effect Transistor (FET) is used as the main switching component of the design and other electronics items are used for suitable application of PWM switching. Figure 6 demonstrates the schematic design of the switch of each motor phase.

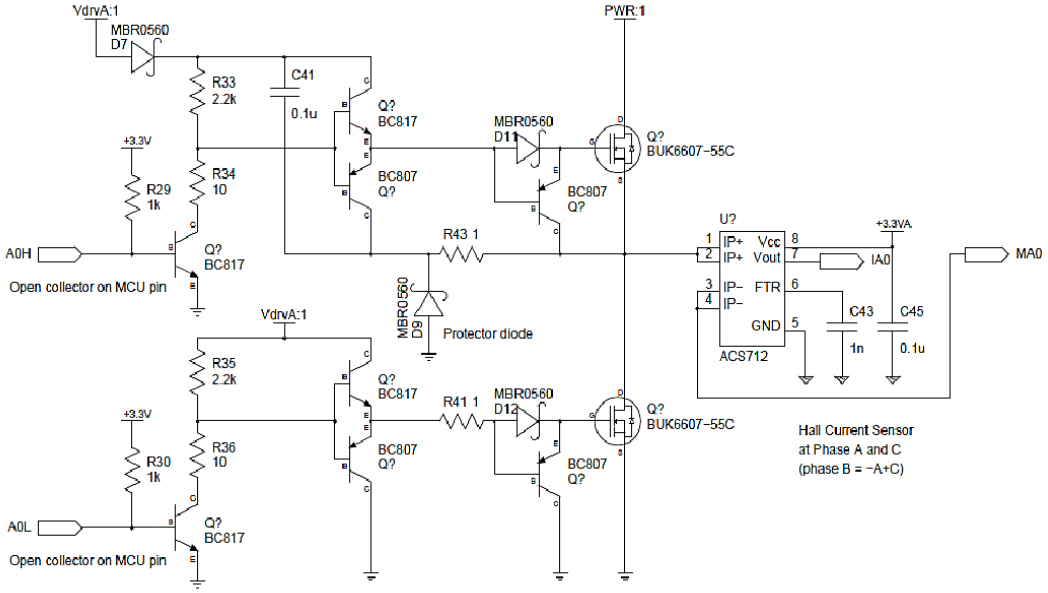


Figure 6: Switching schematic design

The feedback from the rotor position is important for making the flux of the stator to push the rotor with the specific sinusoidal waveform [4]. After the rotor changed position, the flux vector has to change as well. So the interrupt of the controller design has to be used in order to generate the flux vector from the stator according to the update of rotor position. The angle of flux vector which is generated by the stator to the flux vector of the rotor is really important as it limits the torque that is used to push the rotor. In order to get maximum torque, the stator flux vector must be maintained and keep orthogonal to the rotor flux vector. And the inverter board which followed the schematic above by HiveGround as in Figure 7 is used.

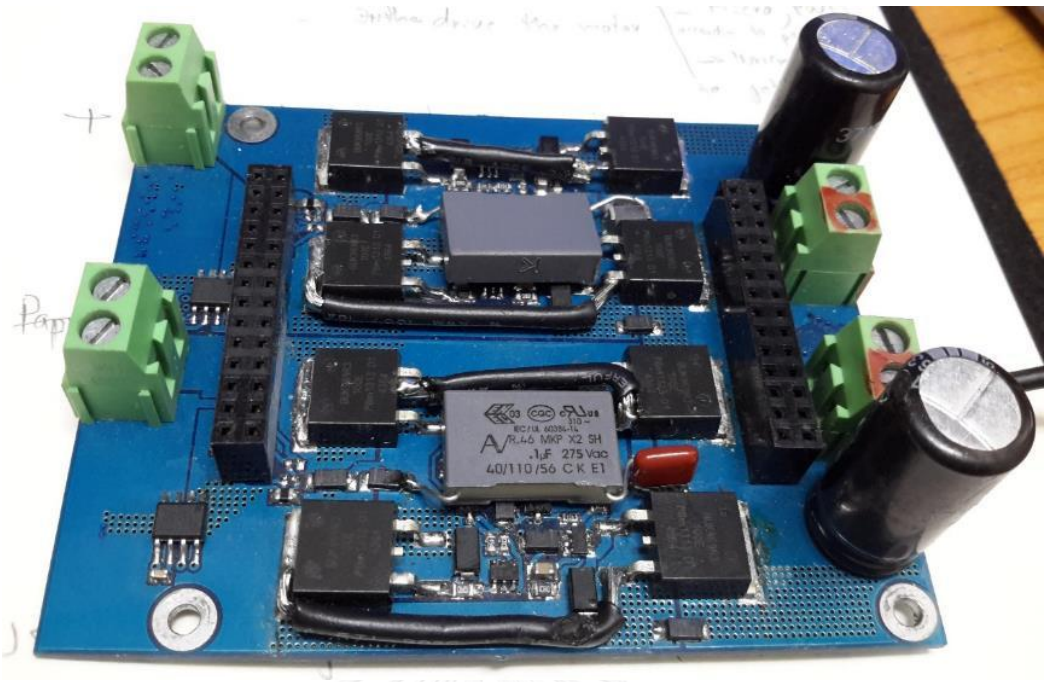


Figure 7: Inverter Driver Board by HiveGround

2.3 RS485

The communication between PC and Microcontroller is needed as the PC is used as the top level of controller of the system. The communication medium must has capacity to communicate a PC to multiple microcontrollers and use just only few wires for this connection. And serial communication RS485 is chosen.

RS485 is a kind of serial communication which enables multi devices to connect to each other using only two signal wires. The transmission data within 32 nodes of this RS-485 can transfer up to the speed of 35Mbps for the distance lest then 12m and 100kbps if the distance is up to 1.2km [6]

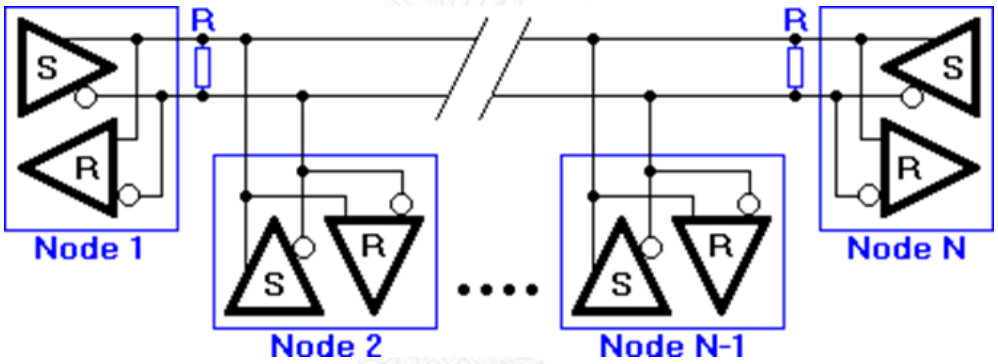


Figure 8: RS485 network topology [6]

In the picture above, the general network topology of RS485 is shown. N nodes are connected in a multipoint RS485 network. For higher speeds and longer lines, the termination resistances are necessary on both ends of the line to eliminate reflections, and 120 Ω resistors are used [7].

As our design MA3485 is being use for making interface between MCU and PC. Figure 9 shows the ping configuration of the MAX3485.

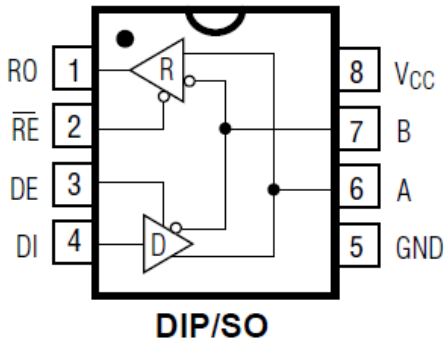


Figure 9: MAX3485 pin configuration

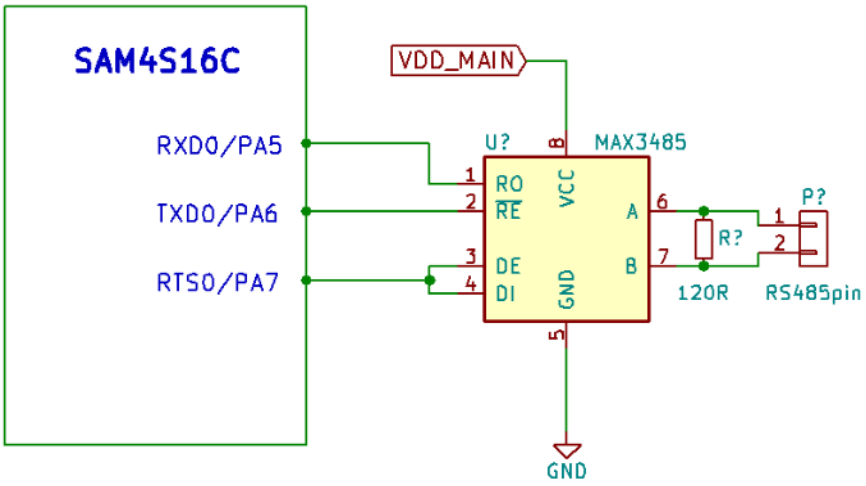


Figure 10: Connection between MAX3485 and MCU

The pin connection between the MCU and MAX3485 is shown in the Figure 9 and the resistance 120 ohm is added to the output pins A and B in order to eliminate the reflection signal.

2.4 Resolver

A **resolver** is a type of encoder which transforms rotary of its rotor to electrical signal in order to measure degree of rotation [8]. Normally it is used for the kind of high precision application of rotation measuring and it also endure to the tough environment, robustness, measurement accuracy, non-contacting working principle and resolution over a wide ambient temperature [9].

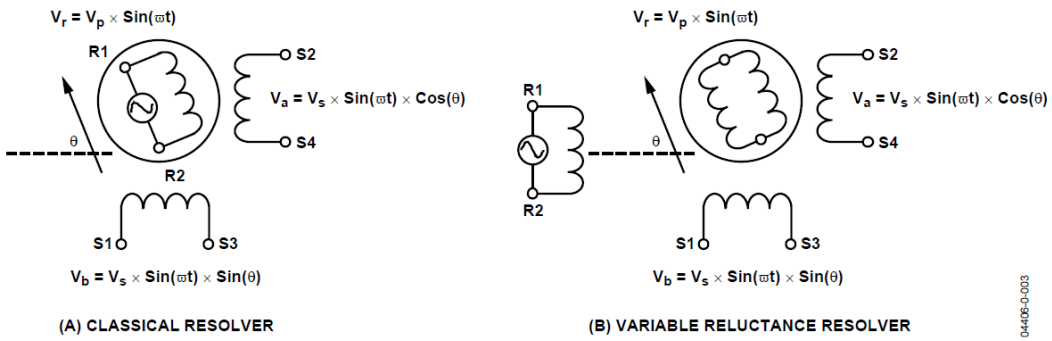


Figure 11: Classical Resolver and Variable Reluctance Resolver [8]

There are two type of resolver Classical Resolver (Figure 11 A) and Variable Reluctance Resolver (Figure 11 B). Classical resolver has a primary winding on its rotor and two secondary winding on the stator. In the case of variable reluctance resolver, the primary and the secondary of the winding are mounted on the stator, but there is a saliency which located in the rotor are designed to provide the sinusoidal variation in the secondary coupling the angle position. Even there are some different designs but these two kinds of resolver generate the same electrical output signal (S3-S1, S2-S4) with the same excitation signal (R2-R1) [8, 10].

When the primary winding (R2-R1) is excited with a sinusoidal signal as in Equation 1, it creates the signal output in the secondary winding (S3-S1, S2-S4). The signals which were induced in the secondary winding is the function of the rotor position to the stator position. As the configuration of the secondary winding which is displace 90^o from each other, the two output sinusoidal signal are phase shift 90^o respect to each other. The relationship of the input and output signal are shown in the Equation

(2), Equation (3), and Equation (4). The Equation (3) is sine signal; Equation (4)(4) is the cosine signal.

$$R1 - R2 = E_0 \sin(\omega t) \quad (2)$$

$$S3 - S1 = T \times E_0 \sin(\omega t) \times \sin(\theta) \quad (3)$$

$$S2 - S4 = T \times E_0 \sin(\omega t) \times \cos(\theta) \quad (4)$$

where:

θ	is the shaft angle
ω	is the angle velocity which is got from the excitation frequency
T	is the resolver excitation ratio
E_0	is the excitation signal amplitude

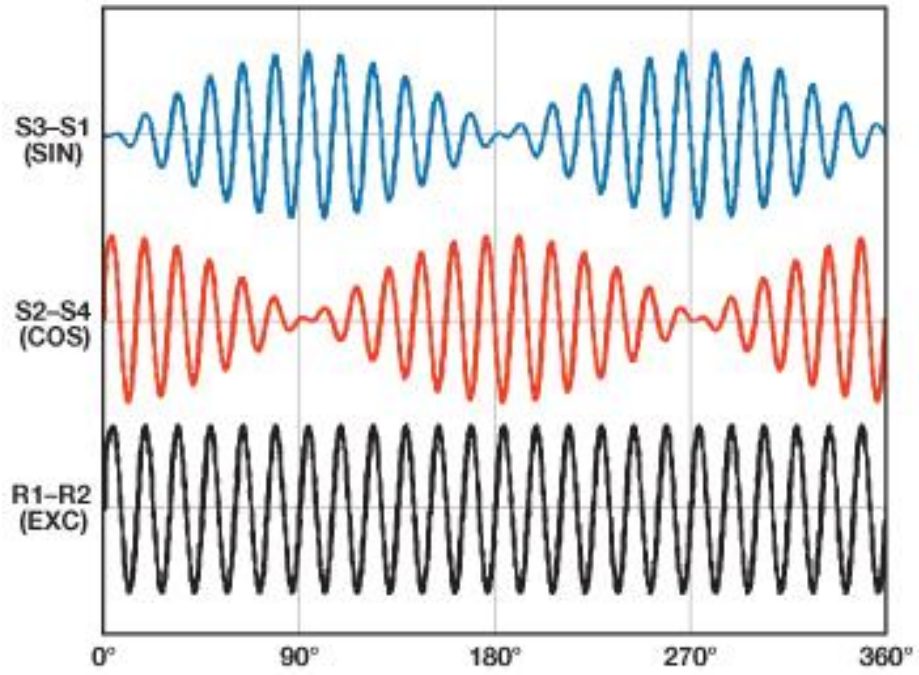


Figure 12: Resolver signals [10]

Figure 12 is the graphical represent of the input signal in the primary winding (excitation signal) and the sine and cosine output signal from the secondary winding of the resolver. It shows that the sine single has the maximum amplitude at 90^o and 270^o, whereas the cosine gets to the maximum amplitude at 0^o and 180^o. This different of the maximum amplitude points is happened due to the configuration of the secondary winding on the stator which is displace 90^o from each other.

The specification of the resolver is really important and must be considered when designing the interface in the application. The most critical of electrical parameter specification are summarized in the table.

Table 2: Resolver Key Parameters [10]

Electrical Parameter	Range	Unit	Description
Input Voltage	3-7	V	Recommended excitation signal amplitude to be applied to resolver primary winding R1–R2
Input Frequency	50-20,000	Hz	Recommended excitation signal frequency to be applied to resolver primary winding R1–R2
Ratio (T)	0.2-1.0	V/V	Ratio between the primary and secondary windings signal amplitude
Input Impedance	100-500	Ω	Input impedance of resolver
Phase Shift	±25	degree	Phase shift between excitation signal applied to primary winding (R1–R2) and sine/cosine signals on secondary windings (S3–S1, S2–S4)

2.5 Microcontroller-Resolver Interface

In this project Resolver is used as the feedback device for the position of the robot joint angle and rotor position of the motor in the join of the robot.

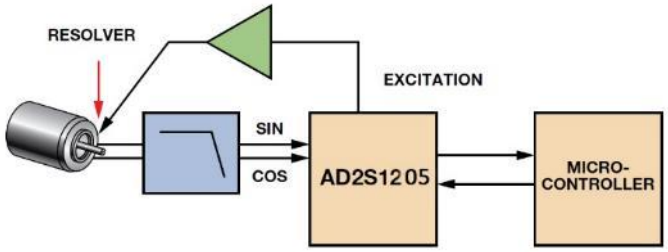


Figure 13: Resolver interface system block diagram

As the output signal from the resolver are electrical signals in the form of sine and cosine, resolver analog to digital device is used in order to translate the output signal from the resolver to digital format before we can use it with the microcontroller and AD2S1205 is used in this application as an interface between microcontroller and

resolver. The block interface diagram of resolver and microcontroller is shown in Figure 13.

AD2S1205 generates excitation signals and inject to the resolver via EXC and $\overline{EXC}$ then resolver output sine and cosine wave form as the output position. These sine and cosine signal are computed by analog to digital decoder AD2S1205 and generates digital output via Serial Peripheral Interface (SPI) for controller. But the excitation signal which was generated by AD2S1205 cannot be injected to the resolver directly, it need to be amplified first.

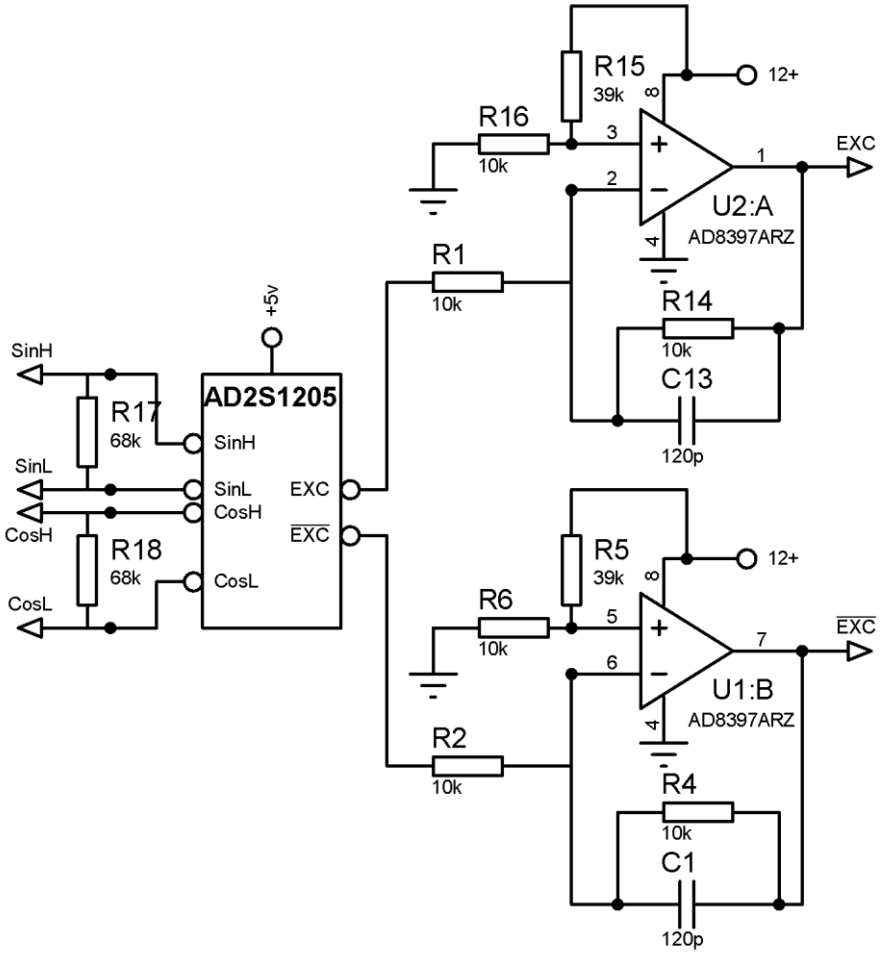


Figure 14: AD2S1205 Interface and Buffer Circuit Based on AD8397 op amp

High-current dual op amp with rail-to-rail output to amplify and level shift the reference oscillator (excitation signal output from AD2S1205 for injecting to the

resolver) AD8397 is used to optimize the interface with the resolver. AD8397 can achieve wide dynamic range, low-distortion, and high output current, make it suitable for the use with resolver[10]. With 310-mA current capability for 32-Ω loads, AD8397 can deliver the required power to a resolver without the use of the conventional push-pull stage. A duplicate circuit gives a fully differential signal to drive the primary winding of the resolver. Available in an 8-lead SOIC package, the AD8397 is specified over the –40°C to +125°C extended industrial temperature range.

Capacitor 120pf and resistor 10KΩ (C1 and C13, R4 and R14 in the Figure 14) form a low-pass filter to minimize the noise of the EXC and $\overline{EXC}$ output. The Figure 14 demonstrates the interface of analog to digital converter for resolver (AD2S1205) and the buffer circuit of the Excitation pins (EXC and $\overline{EXC}$).

2.6 Power Supply of the Controller Board

The controller board also need the power supply to power up the unit. There are 3 main IC that need supply such as MCU SAM4S16C, resolver decoder chip AD2S1205 and Amplifier AD8397. AD9397 which is the amplifier for excitation signals from AD2S1205 be for injection to the solver is use 12V DC which got form outside power supply. The power supply 12V DC is also dropdown by the 800mA low-dropout linear regulator LM1117MPX50 [11] to get 5V DC for supplying resolver-to-digital converter AD2S1205 and Inverter board. As the MCU SAM4S16C need two level of power supply 3.3V and 1.2V [12], so the 5V DC we got from LM1117MPX50 is dropdown by LD1117AS33RT chip to get 3.3V and BU12TD3WGRT chip is used for drop down the power from the 3.3v to 1.2V.

Figure 15 presents the structure used for supplying power to the components needed of the controller board.

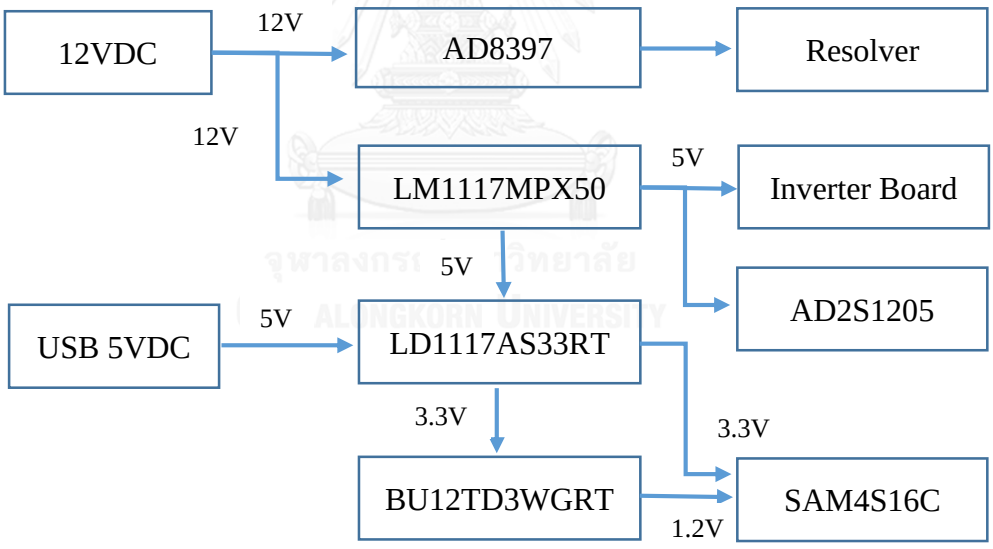


Figure 15: Power Supply Structure

The Figure 16, Figure 17, and Figure 18 show the designing schematics to get the power supply 5v, 3.3v and 1.2v for the supply of the components used in the controller board.

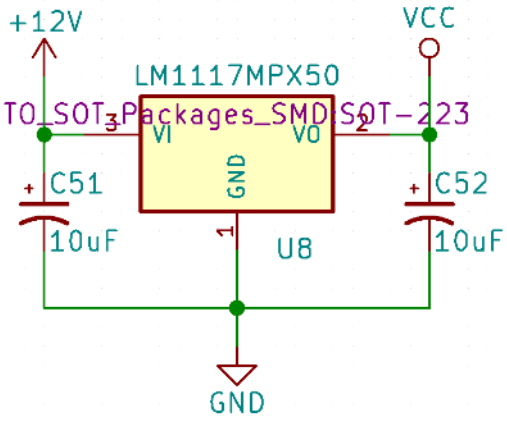


Figure 16: Linear Regulator LM1117MPX50 for 5V Supply

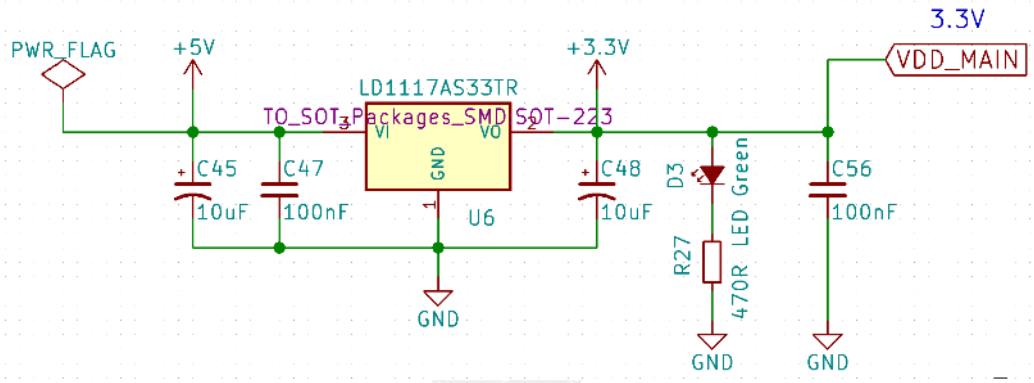


Figure 17: Linear Regulator LD1117AS33 for 3.3V Supply

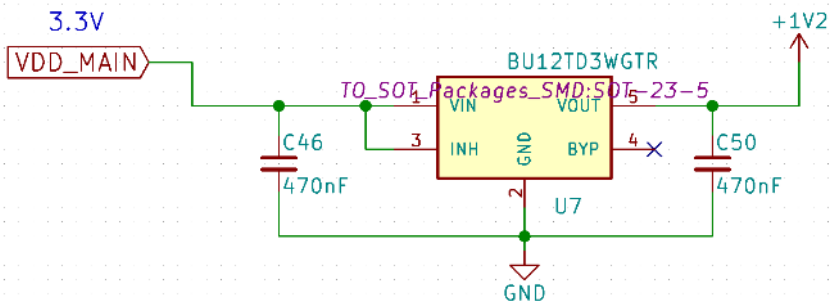


Figure 18: Linear Regulator BU12TD3WGTR for 1.2V Supply

2.7 PCB Design

2.7.1 KiCad

For the PCB design computer-aided drawing software is needed and Kicad 4.0.1 is used in this project for PCB design tools. KiCad is an open source software suite for Electronic Design Automation (EDA) for application of electronic schematic diagrams and PCB drawing. The programs provide PCB Layout, and Schematic Capture with Gerber file needed for production. It can run in many kind of operating system such as OS X, Linux and Windows with the license under GNU GPL v3.

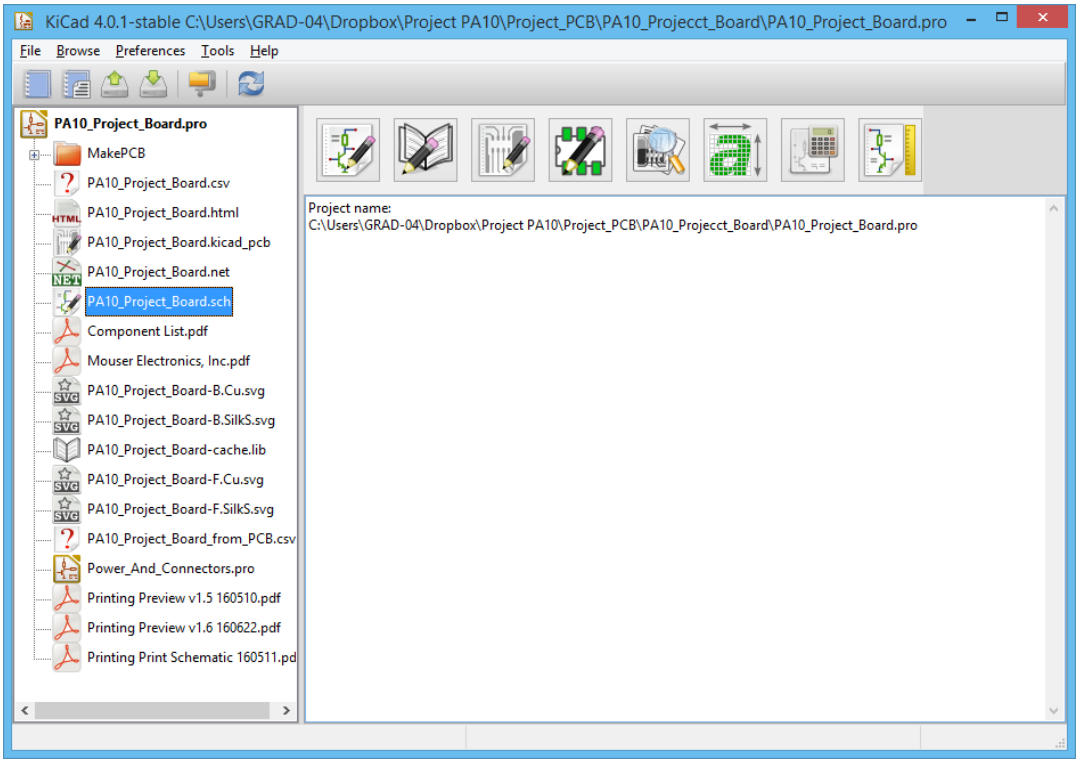


Figure 19: Kicad 4.0.1

KiCad can be considered professional enough for making complex electronic boards. KiCad does not limit the size of the board that you want to design for your application. I can be used to design the board up to 14 technical layers, 32 copper layer. Kicad can export necessary files needed for printed board production such as drilling file, component location files, Gerber files for photo plotters, and more [13]. The two most important part of PCB design is creating schematic as Figure 20 and creating PCB drawing as Figure 21.

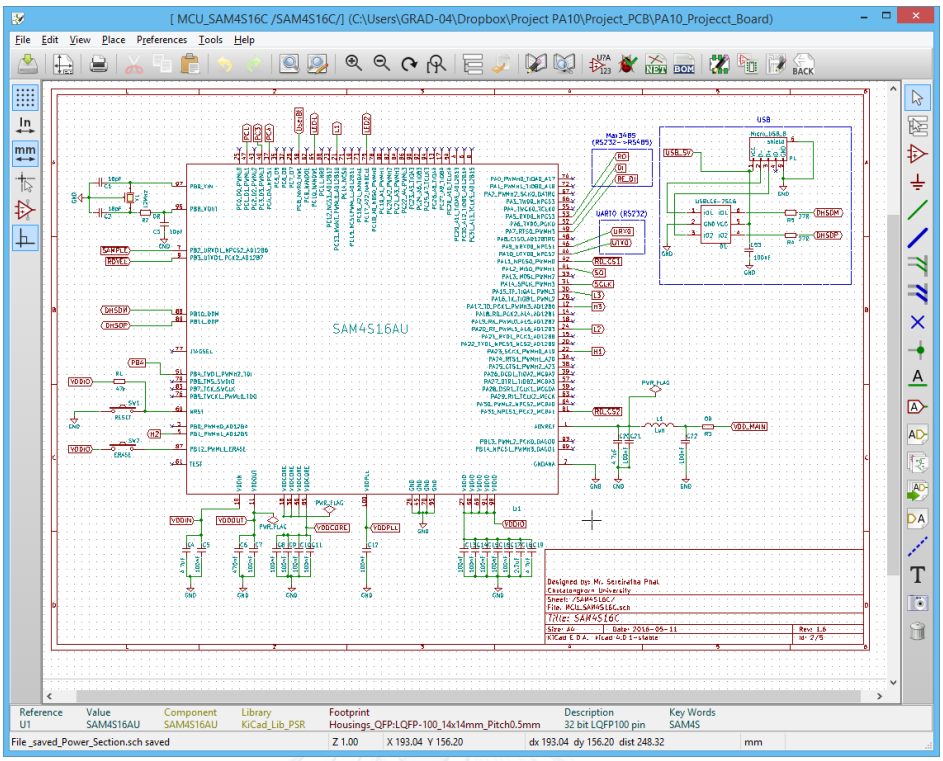


Figure 20: Electronic Schematic Editor

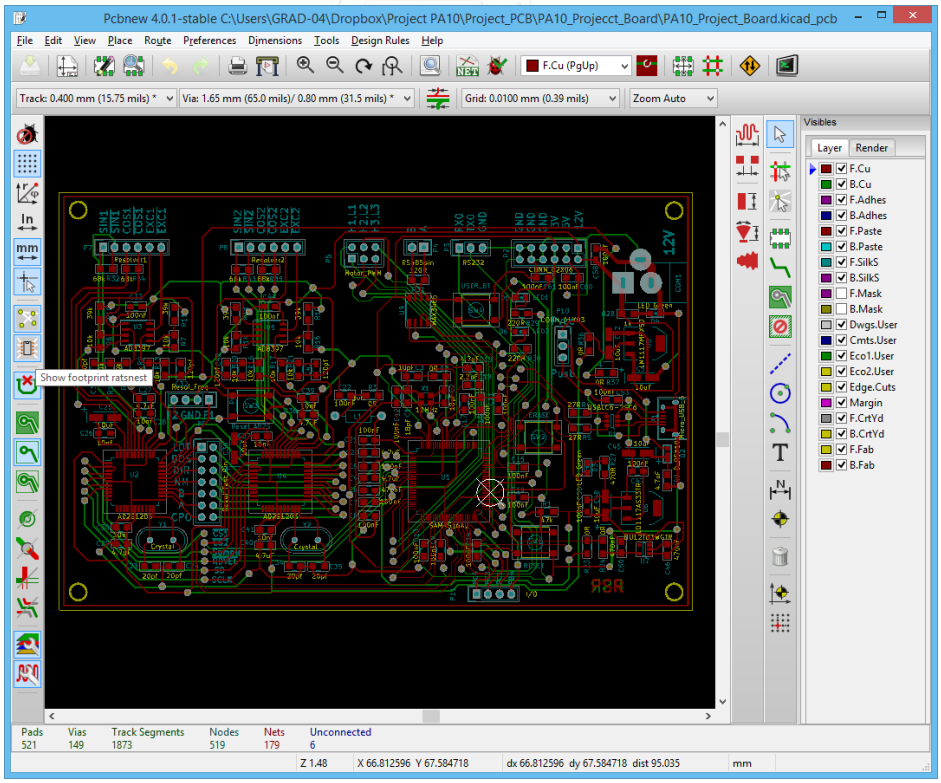


Figure 21: Printed Circuit Editor

2.7.2 PCB Design Workflow

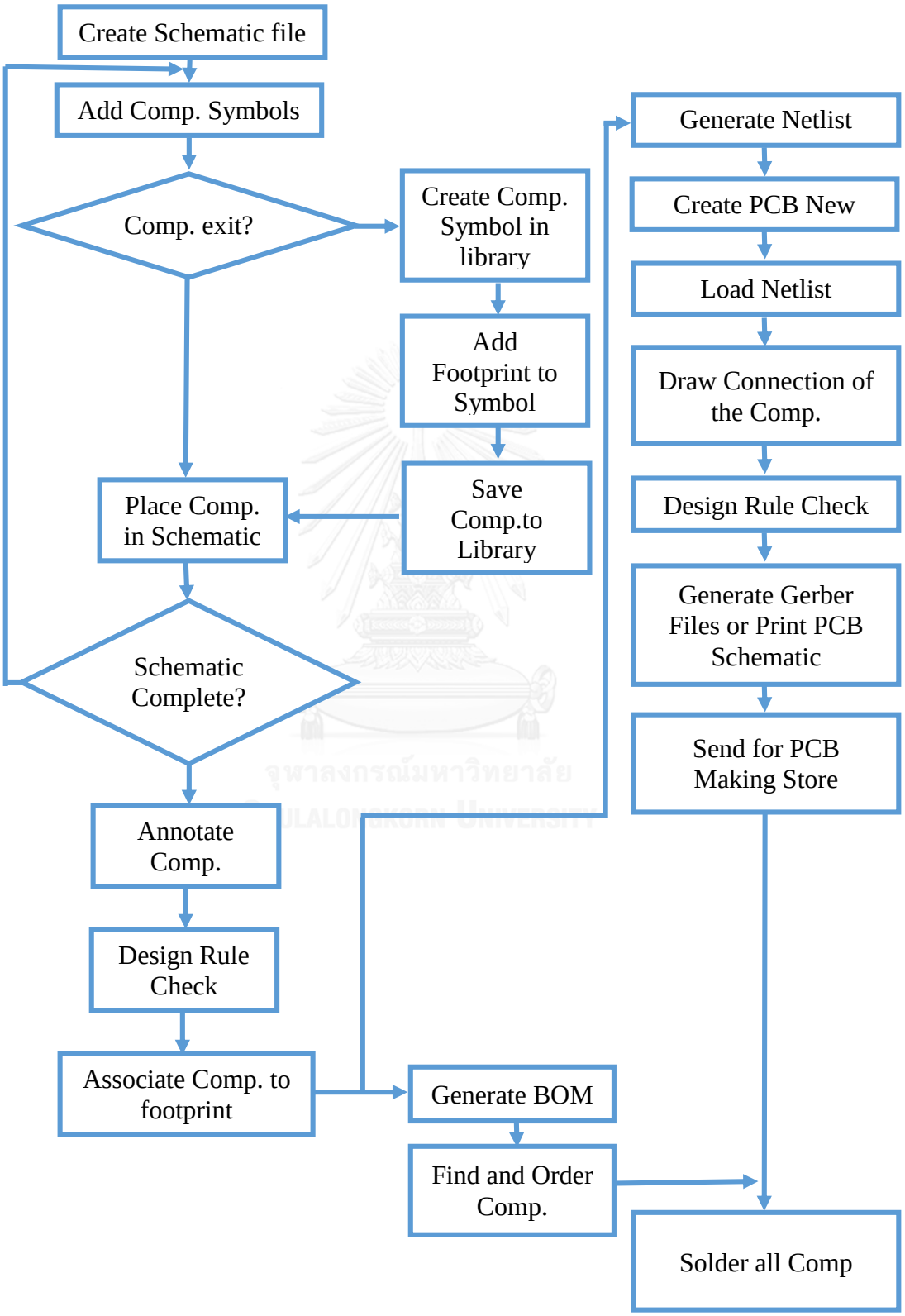


Figure 22: Workflow Design of Making PCB

Figure 22 is the workflow of the PCB design for making the controller board.

For the workflow above we can summary as there are 4 main important part for making the board.

- Firstly is Schematic design which we define the function we want and find the component needed. Next make component connection to each other according the guideline of the components as shown in Figure 20. Sometimes symbols of the component that we want to use does not exist so we have to make it by ourselves before we can use it in the schematic.
- After the first part is finish, we and associate the component symbols that we use with the footprint of the component. Then we can generate bill of material (BOM), the list of component (Figure 23) needed for the design, and order the components in the list for solder it to the board.

Ref	Qty	Value	Footprint
		[C] - Capacitors	
C13 C14 C15 C16 C17 C12 C8 C9 C10 C11 C5 C7 C21 C22 C53 C44 C33 C56 C58 C60 C61 C47	22	100nF	Capacitors_SMD:C_0805_HandSoldering
C18	1	2.2uF	Capacitors_SMD:C_0805_HandSoldering
C19 C4 C20 C27 C29 C38 C40 C49	8	4.7uF	Capacitors_SMD:C_0805_HandSoldering
C6 C46 C50	3	470nF	Capacitors_SMD:C_0805_HandSoldering
C2 C1	2	18pF	Capacitors_SMD:C_0805_HandSoldering
C3	1	10pF	Capacitors_SMD:C_0805_HandSoldering
C32 C31 C43 C42	4	120pf	Capacitors_SMD:C_0805_HandSoldering
C26 C28 C30 C37 C39 C41	6	10nF	Capacitors_SMD:C_0805_HandSoldering
C25 C36 C45 C48 C51 C52	6	10uF	Capacitors_Tantalum_SMD:TantalC_SizeA_EIA-3216_HandSoldering
C23 C24 C34 C35	4	20pF	Capacitors_SMD:C_0805_HandSoldering
		[CON] -	
CON1	1	12V	Connect:JACK_ALIM
		[D] - Diodes	
D1	1	USBLC6-2SC6	TO_SOT_Packages_SMD:SOT-23-6
D3 D4	2	LED Green	LEDs:LED_0805
D2	1	D_RSX101	Diodes_SMD:TUMD2
D5	1	LED1	LEDs:LED_0805

Figure 23: Bill of Materials

- The last part is PCB drawing. Drawing the connection of the components (Figure 21) and send it the PCB store for print it out. Then we can solder the board with the component we ordered.

All the document for this design include board schematic, PCB drawing layout for Making PCB, and bill of matter (BOM) which list all components needed for building this board is show in the appendix.

CHAPTER 3
ALGORITHM AND SOFTWARE DESIGN

3.1 Atmel Studio

Atmel Studio 6 is the integrated development platform (IDP) in the tool provided by Atmel for developing and debugging microcontroller family Atmel Smart, Atmel AVR, and ARM base microcontroller. It supports all Atmel Smart , AVR family MCUs. The Atmel Studio 6 IDP provides you with the easy way to use in code writing environment, debugs and builds your applications code written in assembly, C++ or C code. More than that you can you use it to connect seamlessly to development kits and Atmel debuggers [14].

In this project, Atmel Studio 6 IDP as shown in Figure 24 is being used with all codes are written in C language and Atmel System Framework as the library.

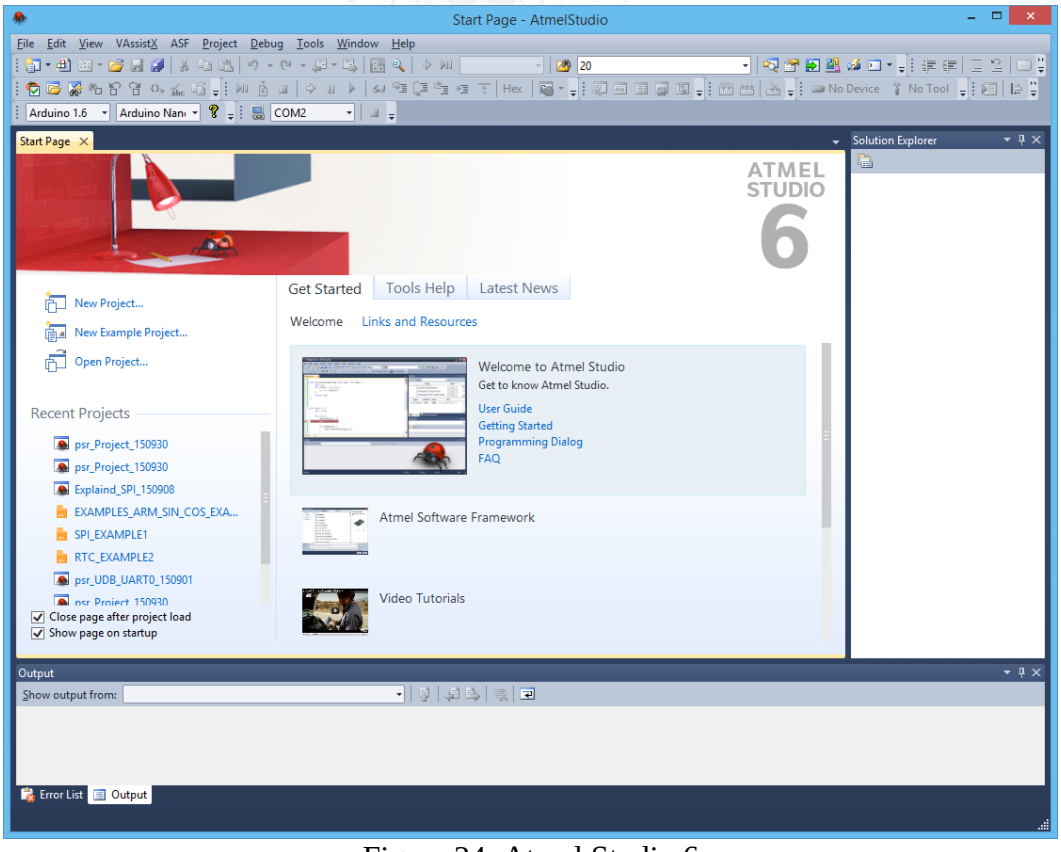


Figure 24: Atmel Studio 6

3.2 Acquire Position and Velocity Output of Resolver from AD2S1205

The MCU reads the angle position from resolver via ADS1205 using serial communication SPI (Serial Peripheral Interface). As the output of AD2S1205 can be can be selected as parallel or serial by $\overline{SOE}$ (Serial Output Enable) pin to high or low. So to enable the data output in the serial form, the low level for $\overline{SOE}$ is hold.

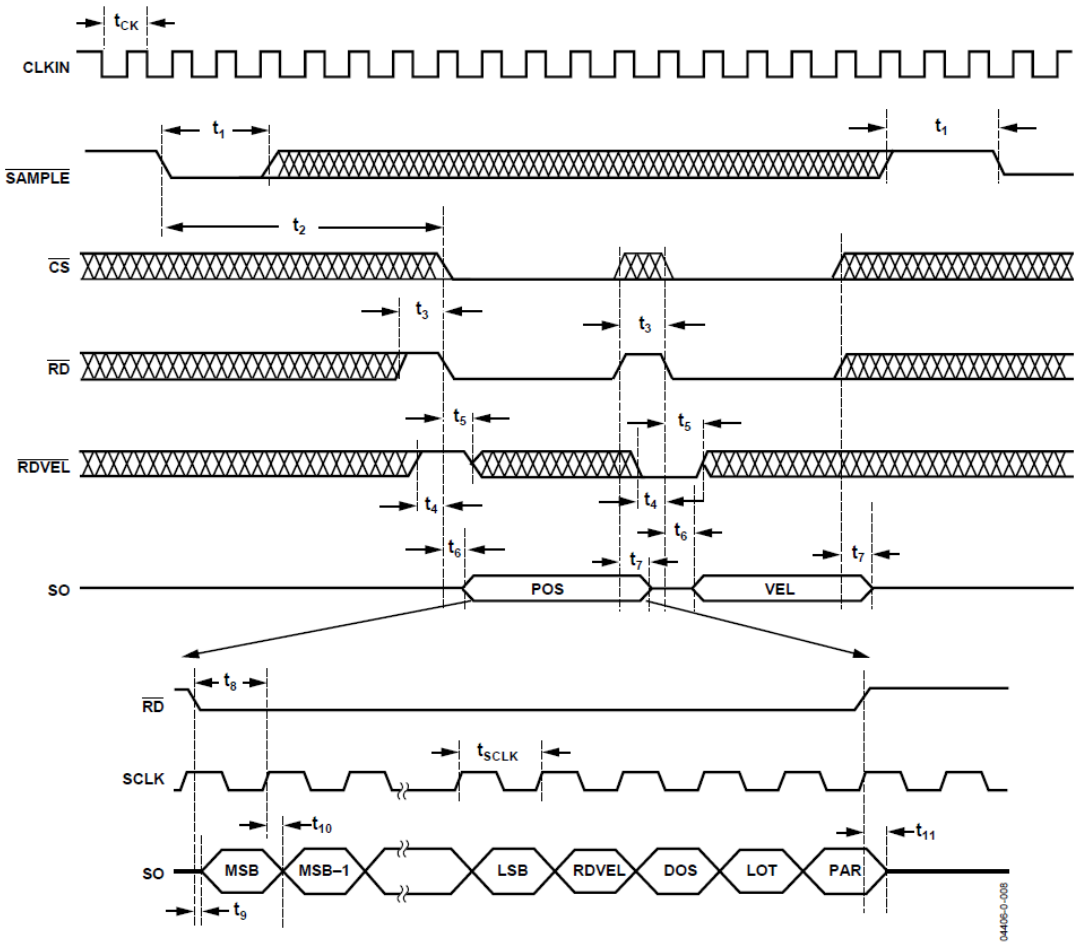


Figure 25: Serial Port Read Timing of AD2S1205 [8]

The Figure 25 shows the configuration and timing diagram of AD2S1205 pins for getting the data, position and velocity.

There are 5 pins needed to control for reading data from the IC in the serial form:

- $\overline{SO}$ slave output, the output data is 16-bit wide. Data is shifted out of the device under the control of the serial clock input, SCLK as shown in the Figure 25. 16-bit word consists of 12-bit of angler data (velocity or position

depending on $\overline{\text{RDVEL}}$ input), one $\overline{\text{RDVEL}}$ bit and three status bits (a parity bit, degradation of signal bit, and loss of tracking bit).

- $\overline{\text{SAMPLE}}$ is for data transfer of position and velocity registers respectively following by high-to-low transition of this pin
- $\overline{\text{RDVEL}}$ is using for selection the type of output from the IC as position when $\overline{\text{RDVEL}}$ is pulled to high level (1) and velocity when $\overline{\text{RDVEL}}$ is pulled low (0).
- $\overline{\text{CS}}$ chip select, is used for SPI communication. The SPI transmits data when this pin us push by high-to-low level.
- $\overline{\text{RD}}$ input is an edge-triggered input that acts as frame synchronization signal and output enable. A falling edge of the $\overline{\text{RD}}$ signal transfers data to the output and data will be available on the output pin SO.
- SCLK, signal clock input for SPI communication

As shown in the Figure 25 we found that he pin $\overline{CS}$ can use the same timing diagram as $\overline{RD}$ so we can. So we can tie the pin $\overline{RD}$ *and* $\overline{CS}$ together and connect to the chip select pin of the MCU. Then we need only 5 pins of microcontroller to read da ta from the resolvers.

Table 3: Timing Diagram of AD2S1205

Parameter	Description	Min	Max
$\mathbf{t_{ck}}$	Clock Period 1/8.192MHz ~ 120ns		
$\mathbf{t_1}$	$\overline{\text{SAMPLE}}$ Pulse Width	2 x t_{ck} + 20n	
$\mathbf{t_2}$	Delay from SAMPLE before $\overline{\text{RD}}/\overline{\text{CS}}$ Low	6 x t_{ck} + 20n	
$\mathbf{t_3}$	$\overline{\text{RD}}$ Pulse Width	18 ns	
$\mathbf{t_4}$	Set time $\overline{\text{RDVEL}}$ before $\overline{\text{RD}}/\overline{\text{CS}}$ Low	5 ns	
$\mathbf{t_5}$	Hold time $\overline{\text{RDVEL}}$ before $\overline{\text{RD}}/\overline{\text{CS}}$ Low	7 ns	
$\mathbf{t_6}$	Enable Delay $\overline{\text{RD}}/\overline{\text{CS}}$ Low to Data Valid		12 ns
$\mathbf{t_7}$	Disable Delay $\overline{\text{RD}}/\overline{\text{CS}}$ Low to Data High Z		18 ns

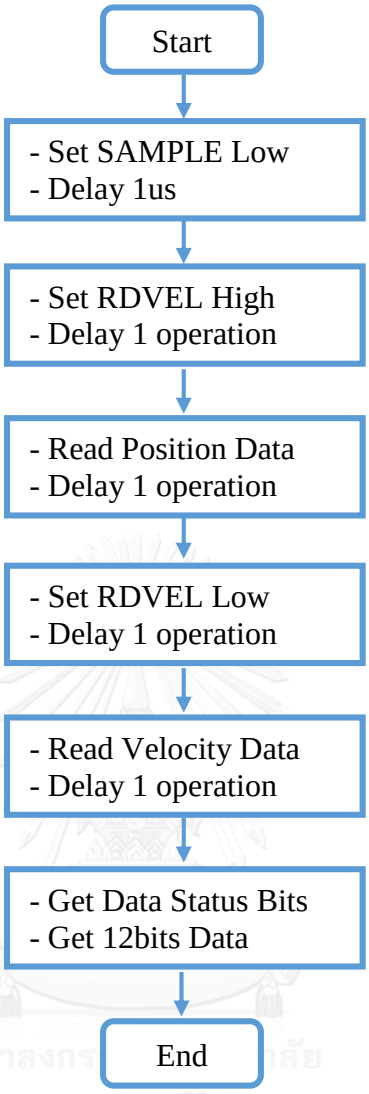


Figure 26: Data Reading Flowchart

The MCU SAM4S16C which we used operates with the speed of 120MHz makes it spends time around 20ns for one operation. Then according to this operation time and the timing in Table 3, we can read the output from the resolver with the process shown in the flowchart of the Figure 26. Below is the code for reading the output data from the resolver via SPI communication.

```
// Read Position and Velocity form AD2s1205-----------------------------
--------------
       //Variable for reading data from Resolver
       # define RESOL_1 0
       # define RESOL_2 1
       uint8_t ps_pcs;
       uint16_t pos;
       uint16_t vel;
       uint16_t _RDVel, _Dos, _Lot, _Par;
```

```
       void AD2S1205_PosVel(uint8_t resol_ID)
       {
       // Read Position and Velocity-------------------------------------
-----
                // Set SAMPLE signal low
                ioport_set_pin_level(SAMPLi,LOW);
                delay_us(1); // delay 1 us
       //++ Get Position
                ioport_set_pin_level(RDVELi,HIGH); // Set RDVEL get
Position
                          nop(); // delay 20 ns

                spi_write(SPI_MASTER_BASE, 0xFF, resol_ID, 1);
                          while((spi_read_status(SPI_MASTER_BASE) &
SPI_SR_RDRF)==0);

                spi_read(SPI_MASTER_BASE, &pos, &ps_pcs); // Read data
from SPI register
                nop(); // delay 20 ns

              //++Get Velocity
                     ioport_set_pin_level(RDVELi,LOW); // Set RDVEL LOW ->
get Velocity
                     nop(); // delay 20 ns

                     spi_write(SPI_MASTER_BASE, 0xFF, resol_ID, 1);
                     while((spi_read_status(SPI_MASTER_BASE) &
SPI_SR_RDRF)==0);
                     spi_read(SPI_MASTER_BASE, &vel, &ps_pcs);// Read data
from SPI register
                     nop(); // delay 20 ns

                      // Set Sample signal High
                      ioport_set_pin_level(SAMPLi,HIGH);

              // Get status bit
                     _RDVel = (pos >> 3) & 1;
                     _Dos = (pos >> 2) & 1;
                     _Lot = (pos >> 1) & 1;
                     _Par = (pos >> 0) & 1;

              // 12 bit data output (Position and Velocity)
                     pos = pos>>4;
                     vel = vel>>4;

       }
```

3.3 Sin Wave Generating Technique

MCU ability of computation is much less than computer nowadays which can perform with high speed up to GHzs. More than this the computation time for floating point variable operation do concern with this low speed and especially non-floating point micro controller unit. It is not only a few clock cycles to do arithmetic operation for floating point calculation, especially multiplication operation which is used very often for controlling process. Some compilers can help making optimization for fixing value variable (Constant) but not the variable which its value is changing from time to time. And there are many variables of this type are frequently used in the programming. So for reducing the calculation time, the optimization of using variables as integer data type must be used and also making the floating point variable to be constant value.

In this project SAM4S16C is used. This MCU can perform up to 120MHz in speed but it is not floating point MCU. The using counter interrupt is necessary for updating the rotor and joint position in a specific period in order to driving the motor. Thus the time for computation in the MCU is limited. The maximum frequency of the PWM of the MCU module is 30KHz [12] so the update SPWM must be at most equal to half of PWM frequency. So the counter interrupt was set to 12.8 KHz or equal to 78μs. In this 78μs period there are many processes to do such as retrieve position from feedback sensors, controlling process and also calculation for SPWM.

One of the most spending time is making SPWM signal which require the value from sin(x) and multiplied by the PWM duty cycle. MCU does not have the operation for Sin(x) and it normally can be computed by using Taylor series:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots \quad (1)$$

but it contains many multiplication operation and it can exceed the limited computation time just for this calculation. As shown in Figure 27, it spent almost 70% for calculation of sin(x) using Taylor series.

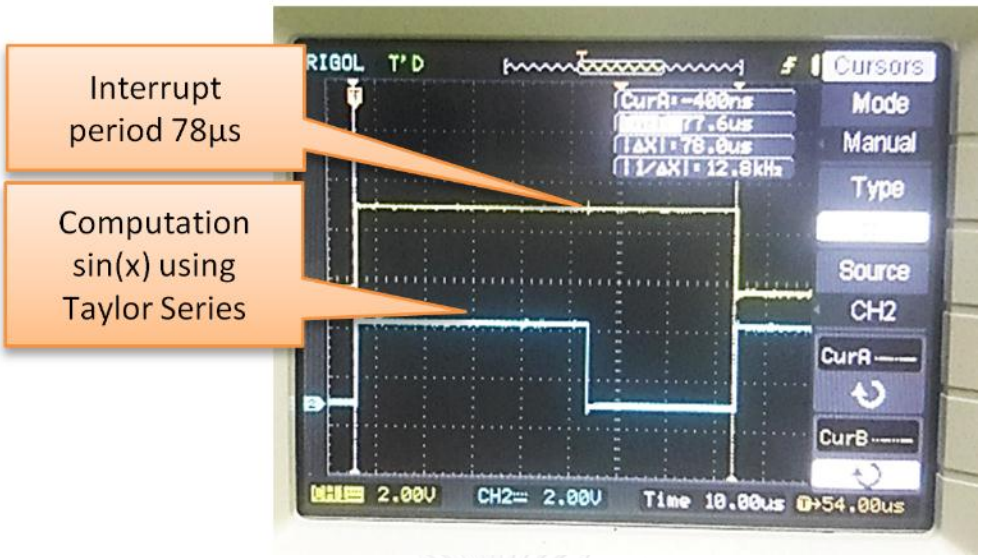


Figure 27: Computation time for Sin(x) using Taylor Series

To solve this problem the use of the constant array for the sin(x) variable is deployed. In the coding, the value of sin(x) from 0º to 360º was divided to 128 resolution and stored as the array elements where the index of the array are the angle x with the conversion 127 resolution as well.

The result that we got from the sin(x) calculation will be converted to duty cycle of the PWM, so we can combine this operation first before store this value in the array. The duty cycle that we use is 1000 resolution so we can combine the operation for duty cycle as the equation (5) and divide it to 128 resolution as Figure 28.

$$Duty = 500\ (1+\sin(x)) \tag{5}$$

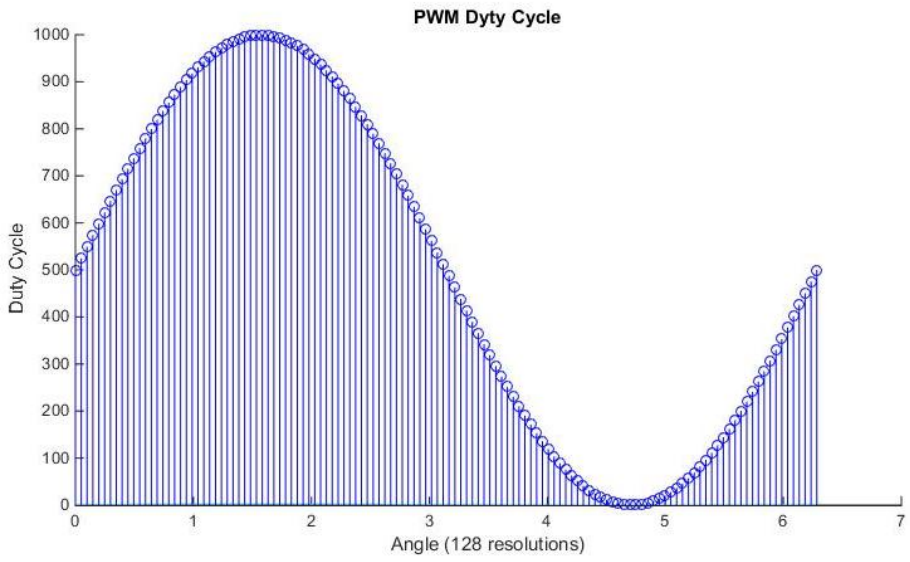


Figure 28: Sin(x) PWM

Then we define the array like this:

```
// Sinx()x = 500+500sin(x)
uint16_t sinx[] = {500, 524, 549, 573, 598, 622, 646, 669, 692, 715, 737,
              758, 779, 799, 819, 837, 855, 872, 888, 903, 917, 930, -
              942, 953, 963, 972, 979, 986, 991, 995, 998, 999, 999, -
              999, 996, 993, 988, 983, 976, 968, 958, 948, 937, 924, -
              910, 896, 880, 864, 846, 828, 809, 789, 769, 748, 726, -
              704, 681, 658, 634, 610, 586, 561, 537, 512, 487, 462, -
              438, 413, 389, 365, 341, 318, 295, 273, 251, 230, 210, -
              190, 171, 153, 135, 119, 103, 89, 75, 62, 51, 41, 31, -
              23, 16, 11, 6, 3, 0, 0, 0, 2, 4, 8, 13, 20, 27, 36,46, -
              57, 69, 82, 96, 111, 127, 144, 162, 180, 200, 220, 241,-
              262, 284, 307, 330, 353, 377, 401, 426, 450, 475, 500};
```

Doing this it means that the multiplication of PWM duty cycle and sin(x) just the operation of integer value and constant floating point value only. After the experiment doing these, the computing time is enough for the limited period. The Figure 29 presents the total time using in driving the robot joint buy using calculation technique above. It also include the time for reading the feedback from resolver, calculation for SPWM, and setting the PWM to inverter board.

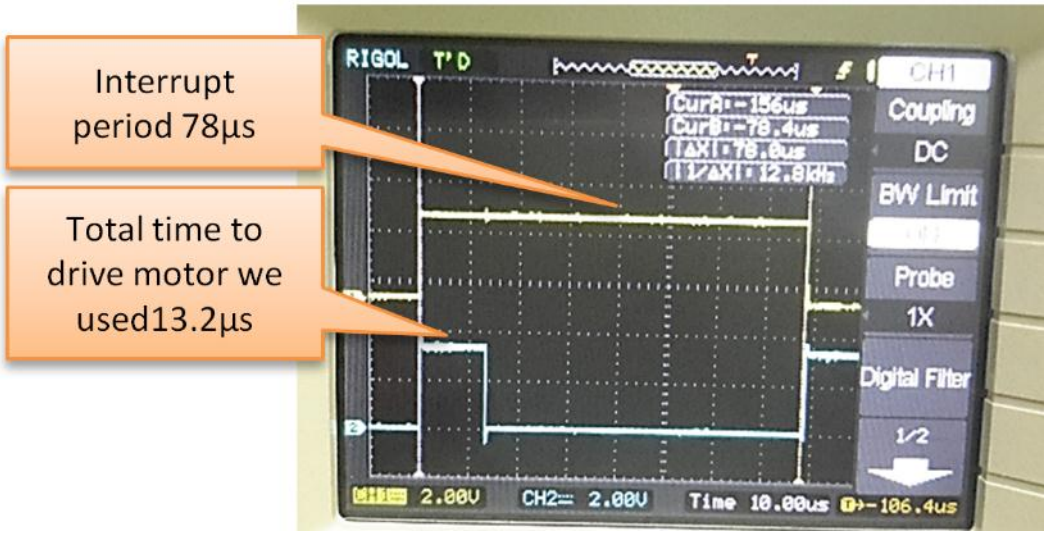


Figure 29: Total time using in driving the joint

3.4 Driving Joint Motor

3.4.1 Theory for Driving

PMSM consists of permanent magnets on its rotor which produces a flux vector. Also when there is current flow through the stator of the PMSM it produces another flux vector as show in Figure 30. This flux vector which created by the stator make the force to push the rotor flux and make the motor rotate.

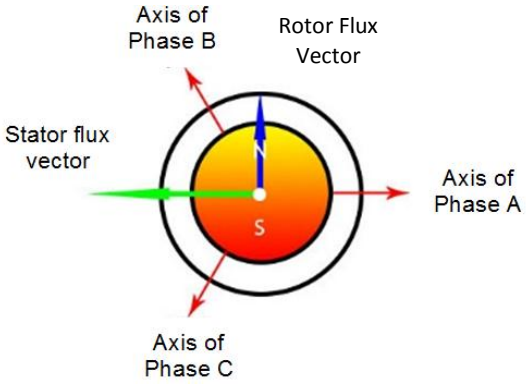


Figure 30: Rotor and Stator Flux Vectors

The angle of the rotor and stator flux vector do matter to the torque of the motor. As the Figure 31[15] shows the relationship between motor toque and the angle. We can see clearly that the optimum of the motor torque is happen at the angle 90^{o} and -90^{o}.

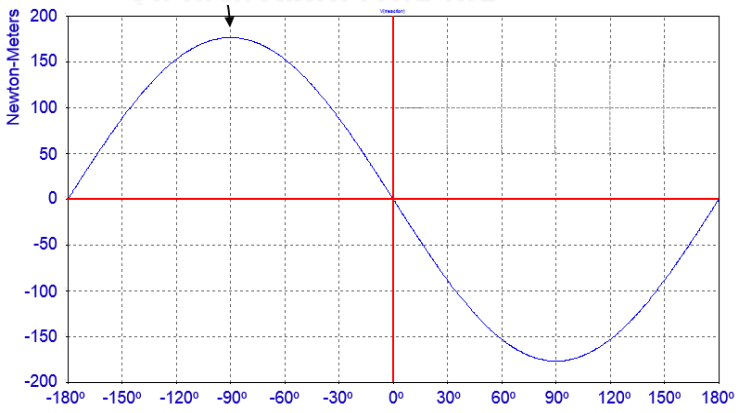


Figure 31: Motor Torque vs Angle between Rotor and Stator Flux Vectors [16]

So we can get the best torque performance out of the motor by keeping the torque of the motor by keep keeping the flux produce by the stator 90^{o} to the flux vector of the rotor. When the motor got the maximum torque the motor also move with the maximum speed.

The problem of keeping this 90^o angle is defining the position of the rotor when it is moving. So after knowing the position of the rotor they can calculate the voltage needed for driving the motor as in equations (6), (7), and (8). There are methods to find $\varphi(t)$ for sensorless as in [17, 18] by feedback the current flow through two phases of the motor. But as our motor joints already have resolver as feedback for the rotor position so we can get $\varphi(t)$ easier.

Phase A: $$Va = \sin(\varphi(t)) \quad (6)$$

Phase B: $$Vb = \sin\left(\varphi(t) + \frac{2\pi}{3}\right) \quad (7)$$

Phase C: $$Vc = \sin\left(\varphi(t) + \frac{4\pi}{3}\right) \quad (8)$$

3.4.2 Experiment and Design

There are experiments for finding the initial angle of the rotor reference to the initial point of resolver. In the experiment we supply the voltage to the 3 phases of the motor with Va, Vb, and Vc; then record the position output from resolver of the rotor and φ for the sinus wave form. The φ ranges from 0 to 127 (which represent 0^o to 360^o) as we use for PWM duty cycle in the section above also the output from the resolver is 0 to 4095 for one round. The program below was used for the experiment.

```
// Test for Initial point of
Point=================================================================

void testFindInitialRotor()
{

	if(ioport_get_pin_level(BT) == LOW) // Push User button to increase
sii 1 resolution
	{
		sii += 1;

		if (sii > 127)
		{
			sii = 0;
		}

printf("Roto Pos: %d, sii: %d, pos-sii: %d \n ",posM , sii, posM/32-sii);

	}

	delay_ms(200);
}
```

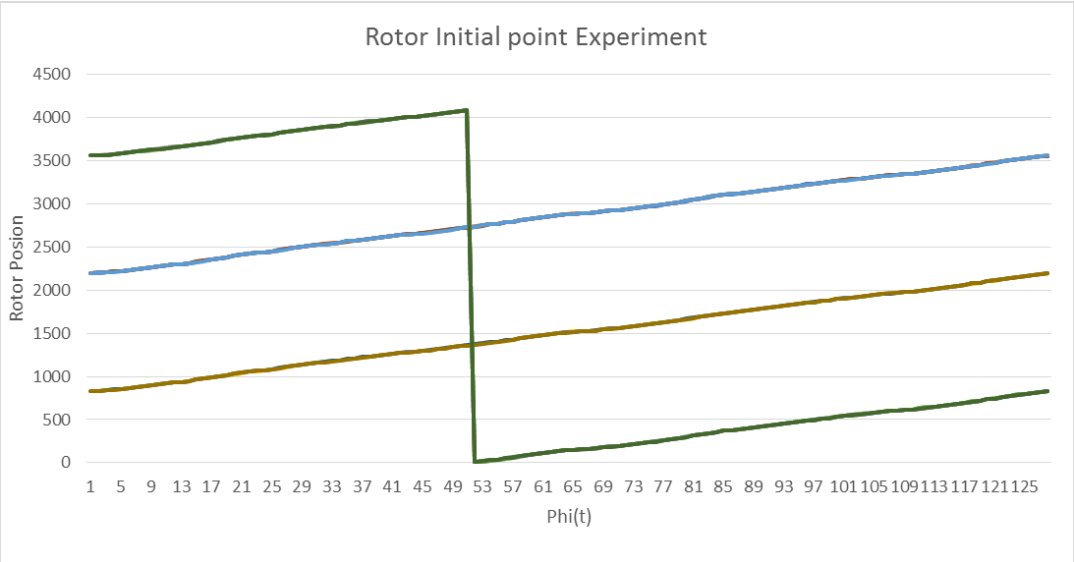


Figure 32: Experiment for finding rotor initial point

Figure 32 shows the experiment result and we can see that 3 rounds of the input signal (supply φ from 0 to 127 for 3 rounds) can rotate the rotor 1 round, also the position of the rotor seem to be linear to the φ . With this result we can determine φ in the function of rotor position, PosM. So we can create equation for defining φ when we know the rotor position from the experiment result above as in equation (9)

$$\varphi = \begin{cases} 0.093\ PosM & for\ 0 \leq PosM < 1365 \\ 0.093\ (PosM - 1365) & for\ 1365 \leq PosM < 2730 \\ 0.093\ (PosM - 2730) & for\ 2730 \leq PosM \leq 4095 \\ 127 + 0.093\ (PosM + 1) & for\ 0 \leq PosM < 1365 \end{cases} \quad (9)$$

where:

φ: is the angle in $\sin(\varphi)$ of the supply voltage of the motor

PosM: is the rotor position of the motor when we supply the voltage of $\sin(\varphi)$

The equation (9) alone is not enough for motor to move with the maximum torque, so we did another experiment to find δ the position of the rotor produce by the supply voltage $\sin(\varphi_{max})$ that give the motor maximum torque or maximum speed. And this $\sin(\varphi_{max})$ is the voltage that we need to supply to the motor in order to the maximum performance. We can get φ_{max} as in equation (10) :

$$\varphi_{max} = \begin{cases} 0.093\ (PosM - \delta) & for\ 0 \leq PosM < 1365 \\ 0.093\ (PosM - 1365 - \delta) & for\ 1365 \leq PosM < 2730 \\ 0.093\ (PosM - 2730 - \delta) & for\ 2730 \leq PosM \leq 4095 \\ 127 + 0.093\ (PosM + 1 - \delta) & for\ 0 \leq PosM < 1365 \end{cases} \quad (10)$$

For the experiment me varied the δ value and supply the motor with the $\sin(\varphi_{max})$ voltage and measure the speed of the motor from the rotor resolver. And after the experiment we can get tow value of δ that make the motor rotate with the maximum speed, one for forward rotation and another one is for backward rotation.

For the forward rotation we got $\delta = 450$, and for backward rotation we got $\delta = 1250$. Then we can implement in the form of code below.

The function below is used to update the angle of the voltage needed to supply the motor in order to drive it forward or backward with the maximum torque we can get. So when we want to drive the motor forward we just need to input the parameter **MoveF** or **MoveB** to the function for driving motor backward.

```
#define MoveF 450    // Fastest speed forward speed= 22.86 V=12v, I=1.13
#define MoveB 1250   // Fastest speed Backward speed= -23.60 V=12v, I=1.03
uint16_t sii = 0;
void driveMotor(uint16_t MoveDir)
       {
              uint16_t delta = MoveDir;

              if (posM>= delta && posM < delta+1365)
              {
                     sii = 0.093 *(posM - delta);
              }
              else if (posM >= delta+1365 && posM< delta+2730)
              {
                     sii = 0.093 *(posM - 1365 - delta);
              }
              else if (posM >= delta +2730 && posM <= 4095 )
              {
                     sii = 0.093 *(posM - 2730 - delta);
              }
```

```
            else if (posM >=0 && posM <= delta)
            {
                    sii = 127 + 0.093*(posM-delta+1);
            }


      }
```

The flowchart in Figure 33 show the flow of driving the motor. After initialize the controller board, we read the position of the rotor. Then calculate the PWM duty cycle needed for generating Va, Vb, and Vc, Next update the PWM output. As we update the PWM, the rotor of the motor rotate to the new position, so this process need to perform again and again with the using of interrupt in MCU.

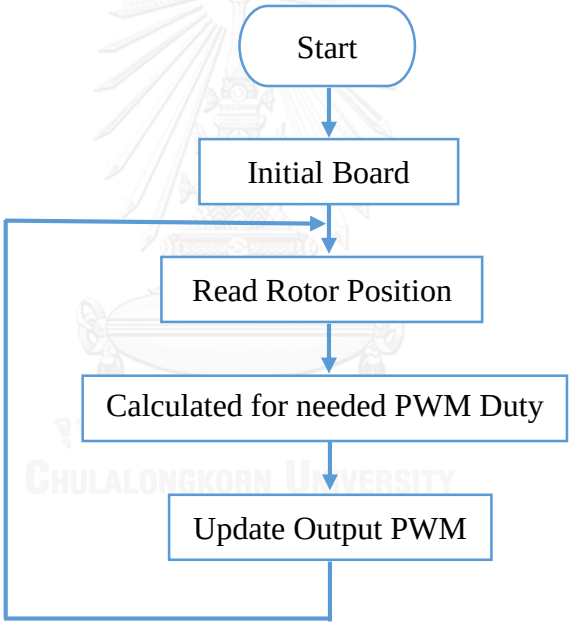


Figure 33: Joint Motor Driving Flowchart

3.5 Joint Controller Workflow

Figure 34 shows the control loop of the joint controller. In this design two close loops are used in order to get the joint of the robot to the desire position. The inner loop used with the aim to drive the motor with the maximum torque and outer loop aim to control the suitable speed and direction of the rotation (forward or backward) to reach the desire set point (position). The Figure 35 presents the controller structure following the design in Figure 34.

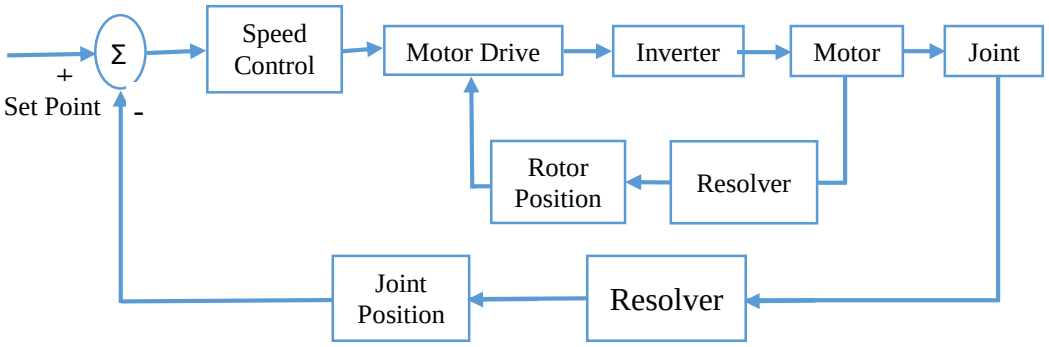


Figure 34: Control System of the Controller

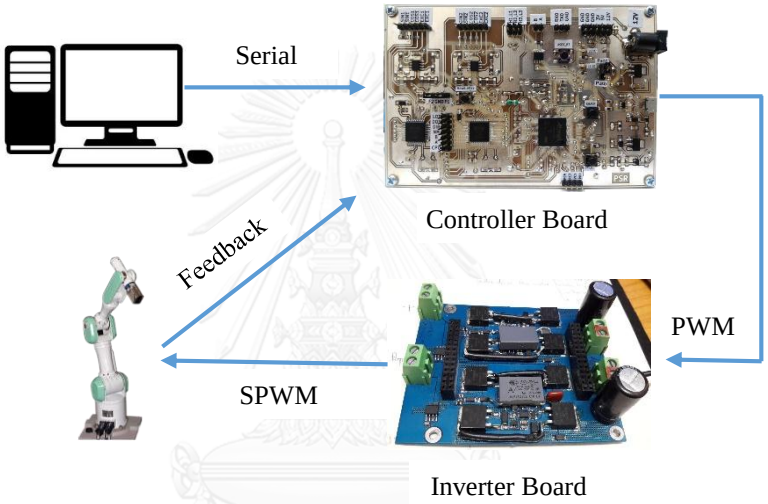


Figure 35: Controller Structure

For the inner loop, at first it reads the position of the rotor by the resolver mounted with the motor rotor to the MCU. The MCU calculates for the suitable PWM cycle as in the section 3.4, then the PWM is updated and sent to the inverter. The inverter supplies the voltage to the motor accordingly PWM in the form of sinusoidal signal.

For the outer loop, it controls the speed and the direction for the rotation. In this loop first of all the feedback of the joint position of the rotor is read by the resolver mounted with the robot joint. Then this feedback position is compared with the set point by the MCU. The error of the set point and the current position of the join is used for calculating the suitable speed for driving the motor and also deciding which direction the motor should move, forward or backward, in order to get closer to the desired position.

```
// Motor control following the set point----------------------------------
void setPoint(uint16_t _desPosition)
{
	int16_t deltaPos;
	deltaPos = _desPosition - posG;

	if (deltaPos == 0)
	{
		setMotorSpeed(0);
	}
	else if (deltaPos > 0)
	{
		driveMotor(MoveF);
		if (deltaPos<150) // decrease speed
		{
			setMotorSpeed(0.45*abs(deltaPos)+30);

		}else{
			setMotorSpeed(100);
		}

	}else{
		driveMotor(MoveB);
		if (deltaPos>-150) // decrease speed
		{
			setMotorSpeed(0.45*abs(deltaPos)+30);

		}else{
			setMotorSpeed(100);
		}
	}
}
```

The programming code above is used for setting up the velocity profile of the motor. We want the speed of the motor to get into the maximum speed in a short time and maintain the speed till the error between the joint position and set point decrease within 150 point of error, we start decrease the speed of the motor as well. When the error equal to zero the motor speed is set to zero.

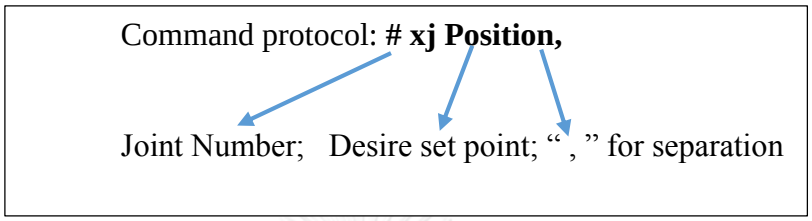
The program code below showing the controller flow in programming.

```
// TC_Handler Interrupt Counter every 78us/12.8KHz-----------------------

void TC_Handler(void)
{
```

```
        volatile uint32_t status = tc_get_status(TC0,0);
        if (status & TC_SR_CPCS) {

        AD2S1205_PosVel(RESOL_M); // Get motor Position and Velocity
        AD2S1205_PosVel(RESOL_G); // Get joint Position and Velocity

        setPoint(desirePos); // set desire position or Set point

        }
}
// Motor Control (Output PWM and Speed Control)----------------------------
void setMotorSpeed(uint16_t _mSpeed )
{
        // Update PWM Dyty Cycle
        pwm_channel_update_duty(PWM, &ch_setting[0], _mSpeed*(1000-
get_sinx(sii))/100 ); // Phase A
        pwm_channel_update_duty(PWM, &ch_setting[1], _mSpeed*(1000-
get_sinx(sii + 43))/100); // Phase B = Phase A + 120 degree = Phase A +43
(43=120degree)
        pwm_channel_update_duty(PWM, &ch_setting[2], _mSpeed*(1000-
get_sinx(sii + 85))/100); // Phase C = Phase A + 240 degree = Phase A +85
(85=240degree)
}

// Sinx()x = 500+500sin(x)
uint16_t sinx[] = {500, 524, 549, 573, 598, 622, 646, 669, 692, 715, 737, -
                758, 779, 799, 819, 837, 855, 872, 888, 903, 917, 930, -
                942, 953, 963, 972, 979, 986, 991, 995, 998, 999, 999, -
                999, 996, 993, 988, 983, 976, 968, 958, 948, 937, 924, -
                910, 896, 880, 864, 846, 828, 809, 789, 769, 748, 726, -
                704, 681, 658, 634, 610, 586, 561, 537, 512, 487, 462, -
                438, 413, 389, 365, 341, 318, 295, 273, 251, 230, 210, -
                190, 171, 153, 135, 119, 103, 89, 75, 62, 51, 41, 31, -
                23, 16, 11, 6, 3, 0, 0, 0, 2, 4, 8, 13, 20, 27, 36,46, -
                57, 69, 82, 96, 111, 127, 144, 162, 180, 200, 220, 241,-
                262, 284, 307, 330, 353, 377, 401, 426, 450, 475, 500};

// if the angle > 127 so change it to 0->127-------------------------------
static uint16_t get_sinx(uint16_t ang)
{
        if (ang>=127){
                ang = ang-127;
        }
        return sinx[ang];
}
```

3.6 Command from Matlab

In order to send position from the PC to the controller board we create a simple Matlab GUI for sending the command. We also make a simple protocol so if we use it with multiple controller board it can know exactly which joint we want to send the command to.



Example: #1j2500, this mean we command joint number 1 to the position 2500. The protocol that we used consist of joint number follow by a letter ‘ j ’ then the position, and finish by ‘ , ’. letter ‘ j ’ is used for separate the joint number and the position command. And ‘ , ’ is used for termination of the command. So when we send the command to the controller board the controller board can know which part is joint number and which part is the desire position.

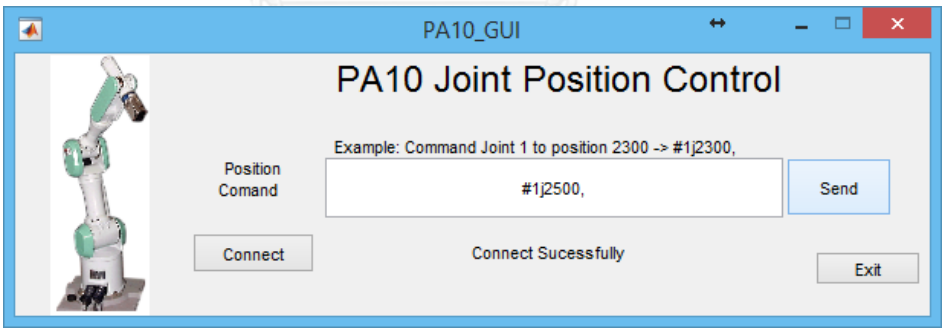


Figure 36: Matlab GUI for Sending Command

The Figure 36 shows the GUI that we made in Matlab for sending position command the controller board. The controller will not recognize command if there is wrong format in the sent command.

CHAPTER 4
CONTROLLER DESIGNING RESULT AND DISCUSSION

4.1 Controller Board

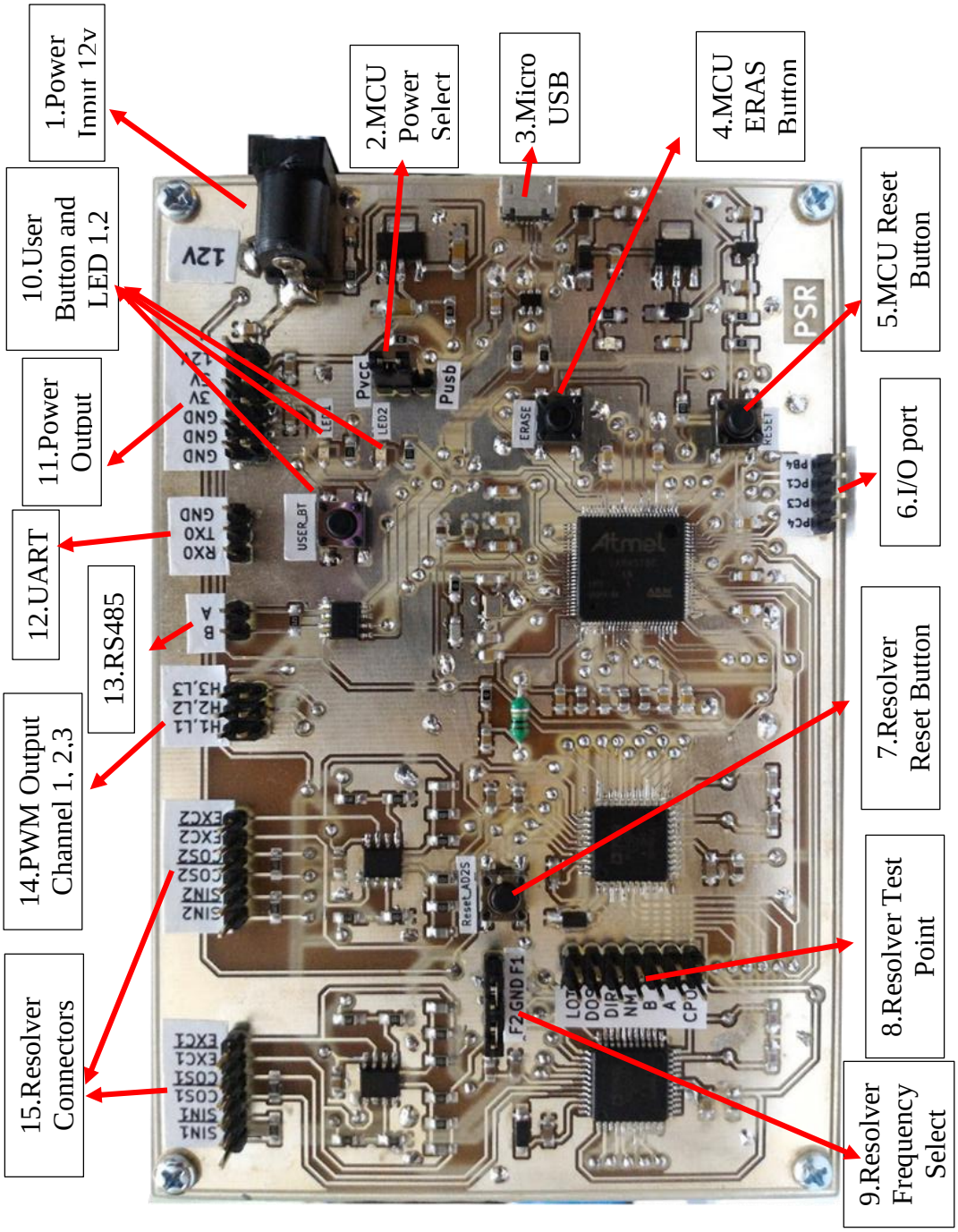


Figure 37: Joint Controller Board

Figure 37 shows the joint controller board which is made according to the schematic design and PCB design by using KiCad.

Feature of the joint controller board:

1) Board power supply 12V
2) MCU Power Selection: for selecting power to supply MCU. The power supply to the MCU can be supplied by UBS or by the main input power according to the selection of the jumper.
3) Micro USB: for programing the MCU via SAM-BA software.
4) MCU ERAS Button: erase code in the MCU.
5) MCU Reset Button: reset MCU.
6) I/O port: 4 general purpose input/output pins.
7) Resolver Reset Button: reset the resolver ICs.
8) Resolver Test Point: tested points for resolver ICs.
9) Resolver Frequency Select: Excitation frequency selection of the resolver IC (Table 4) can selected in 4 frequency levels, and we use 20kHz in this project.

Table 4: Excitation Frequency Selection

Frequency Selection (kHz)	FS1	FS2
10	1	1
12	1	0
15	0	1
20	0	0

10) User Button and LED1,2 : there are 1 button and 2 LED for general purpose usage for user.
11) Power Output: output power pins for additional usage. There are 2pins for 12v output, 2pins for 5v output, 2pins for 3.3v output, and 6pins for GND.
12) UART: for RS232 serial communication.
13) RS485: RS485 serial communication is used for network communication with other board or with PC (one to many serial communication).
14) PWM Output Channel 1,2,3: output of PWM signal for driving 3 phases motor purpose.
15) Resolver Connector Channels 1,2: connector port to resolvers.

4.2 Upload Software to the Board

4.2.1 SAM-BA Mandatory Software Setup

SAM-BA (Boot Assistant) is one of the tools provided in Atmel AT91 ISP (In-System Programming) solution. By using SAM-BA we can program the AT91 family microcontroller easily with the graphical user interface or commend-line interface that it provided. It is also possible to make powerful scripts that can then can be run through the commend line, allowing the automation of many jobs. Those scripts can be written by the programmer or documented through the GUI.

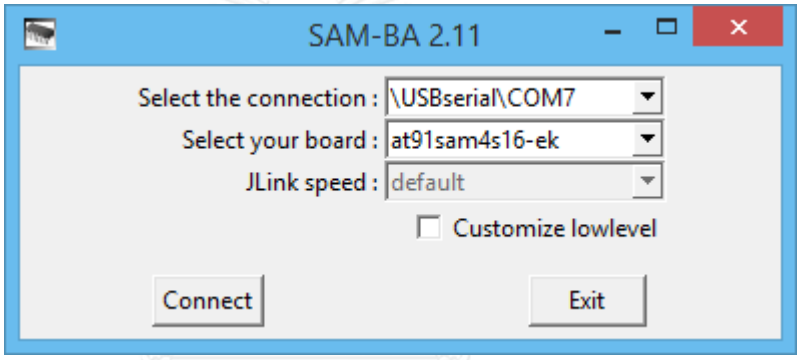


Figure 38: SAM-BA 2.11

The Figure 38: SAM-BA 2.11Figure 38 show the startup of SAM-BA 2.11 and the software can be downloaded via this link: http://www.atmel.com/tools/atmelsam-bain-systemprogrammer.aspx[19].

As our board uses SAM4S16C as the MCU so after start SAM-BA program, so we need to select at91sam4s16-ek at the “Select your board”.

4.2.2 USB Driver Checking

USB driver checking process in summary is shown in the Figure 39.

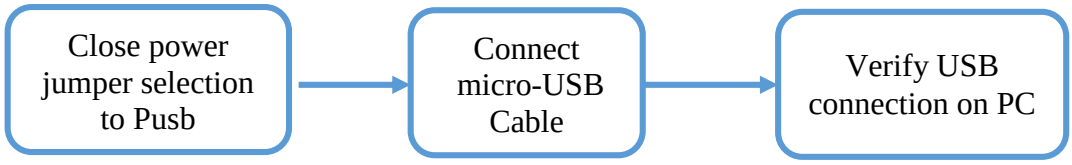


Figure 39: Summary of USB driver checking

Process detail:

1. Choose the jumper at the “MCU Power Selection” to connect between the middle pin and Pusb pin.
2. Connect the board to PC via micro-USB
3. Open “Computer Management” in PC, then choose “Device Management”. Next verify that the USB connection is established, you can see “Bossa Program Port (COMxx)” in Port (COM & LPT) category.

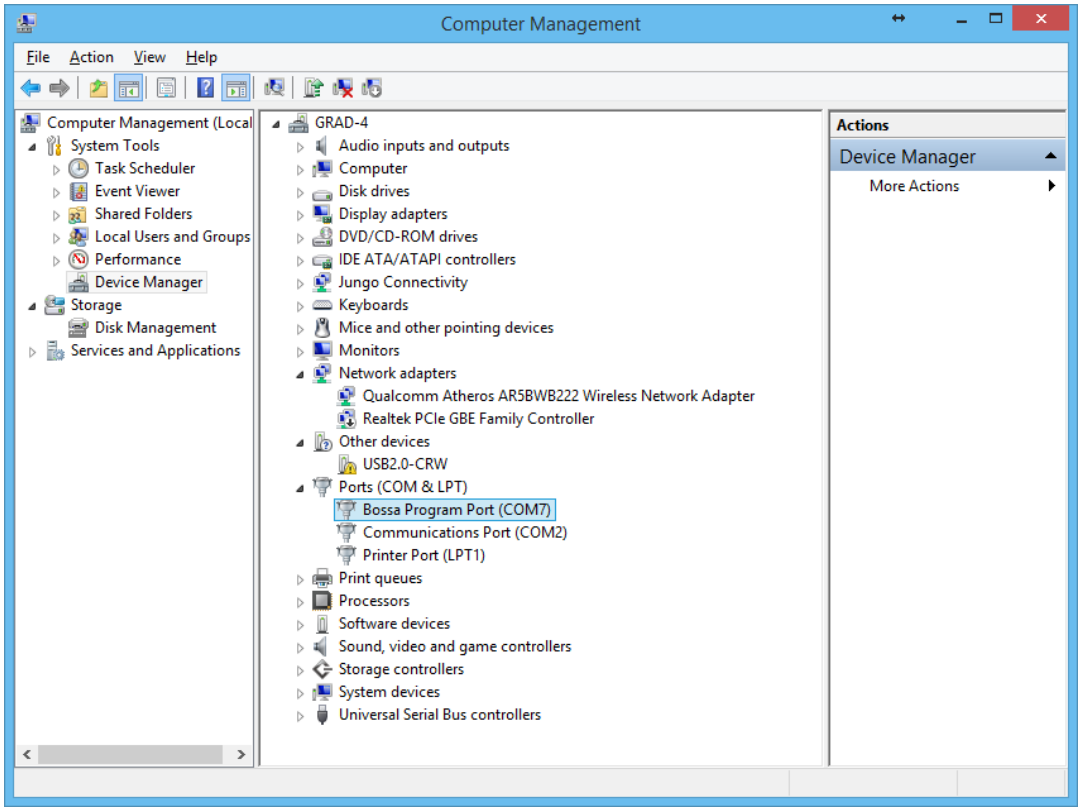


Figure 40: Verify USB driver in Computer Management

4.2.3 Software Upload

To upload the software to the controller board we need the follow the steps as shown in the figure below.

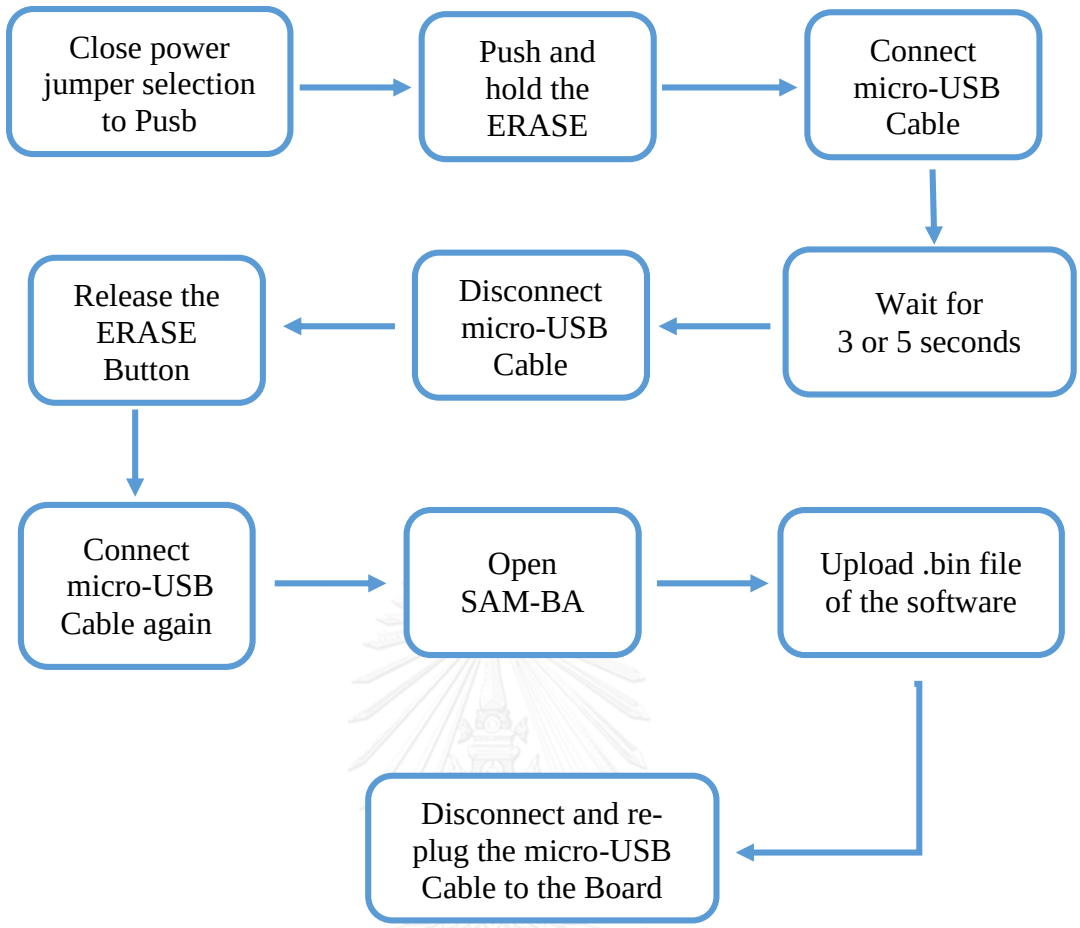


Figure 41: Upload Software Process

Process detail:

1. Choose the jumper at the “MCU Power Selection” to connect between the middle pin and Pusb pin.
2. Push and hold ERASE Button for
3. Connect the board to PC via micro-USB
4. Wait for 3 or 5 seconds
5. Disconnect micro-USB between PM and the controller board
6. Release the ERASE Button
7. Connect the controller board and PC via micro-USB
8. Open SAM-BA, choose the connection port of the controller board port. Sometime it also shows other serial port (COM port) that is not the controller board port, so choose the one that belong to the controller board. Then click connect as shown below.

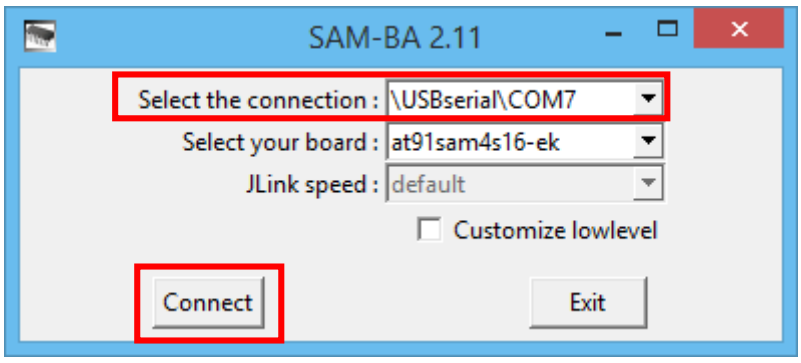


Figure 42: Connect the Controller board in SAM-BA

9. Find and open the software file that we got after compiling our code with the extension .bin and click send file.

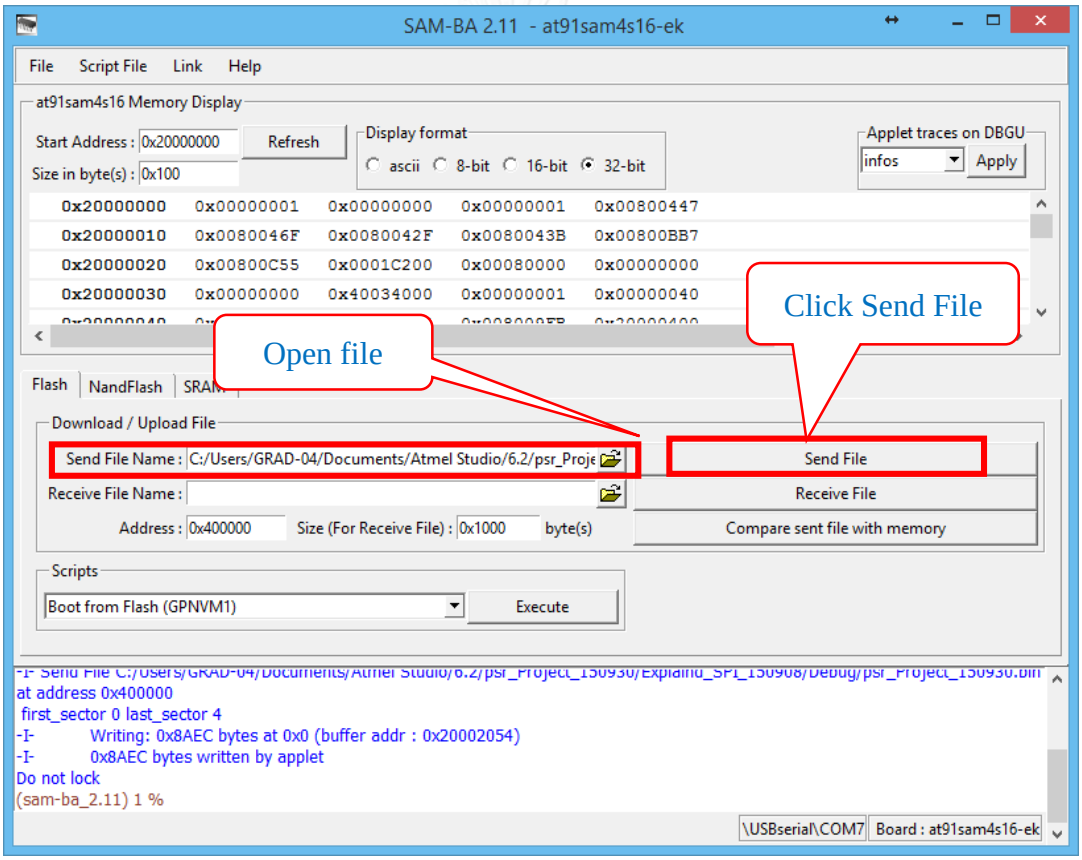


Figure 43: Upload .bin File

10. Disconnect and re-plug the micro-USB cable to the board to rest and the program will start running

4.3 Expense Cost

In order to build the controller board with the following of the schematic design, it use 126 items with 51 types of components.

Table 5: Expanse for Controller Board Production

Items	Quantity	Unit Price (Baht)	Sub Total (Baht)
MCU (SAM4S16C)	1	415	415
Crystals 12MHz	1	25	25
Resolver (ICAD2S1205)	2	838	1676
Crystals 8.19MHz	2	20	40
Amplifier (AD8397)	2	216	432
Regulator 5V (LM1117MPX50)	1	17	17
Regulator 3.3V (LD1117AS33RT)	1	15	15
Regulator 1.2V (BU12TD3WGTR)	1	25	25
Max3485	1	130	130
Diode	1	2	2
ESD protection diodes	1	12	12
Push Button	4	2	8
Micro USB Connector	1	10	10
LED	4	2	8
Connector JACK_LLIM	1	10	10
Connector Pin Header Straight 2 row40	1	12	12
Connector Pin Header Straight 1 row20	1	12	12
Capacitor	57	2.5	142.5
Resistor	34	1	34
Inductor	1	2	2
PCB Publication	1	225	225
Other small expense	1	200	200
Total			**3452.5**

4.4 Experiment and Result Discussion

The Figure 44, Figure 45 and Figure 46 show the performance of the robot joint moving forward to the reach the set-point position. With the precision of the resolver decoder we can get 12bits of position resolution for 360º it mean one resolution output form resolver is 0.088º. As we do experiment of the controller board, in Figure 45 show that we can achieve the exact set point position, 2000(175.78º), from the initial position 321(28º) in 1.813s as the sampling time in the experiment is 2.64ms.

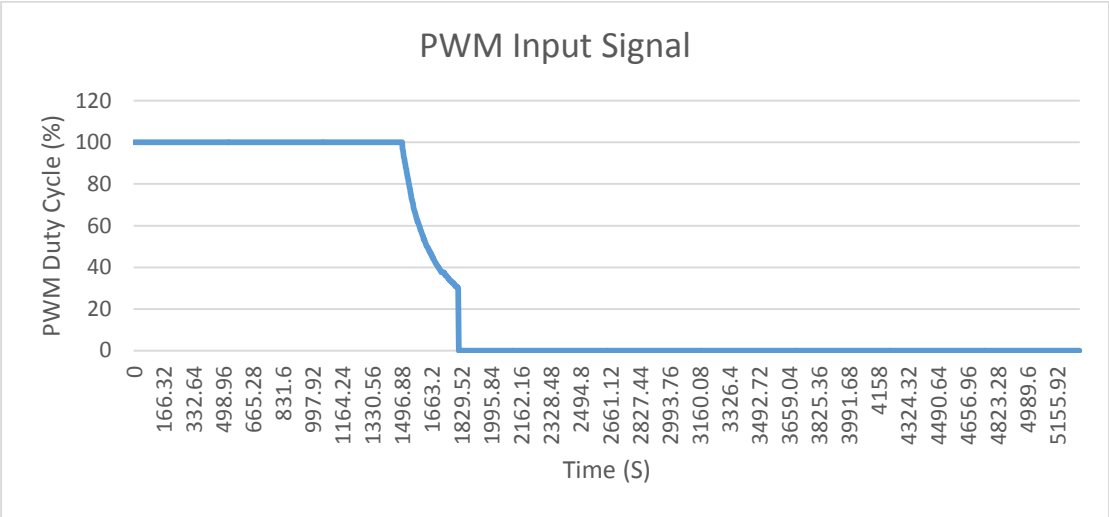


Figure 44: PWM Input Signal When Moving Forward

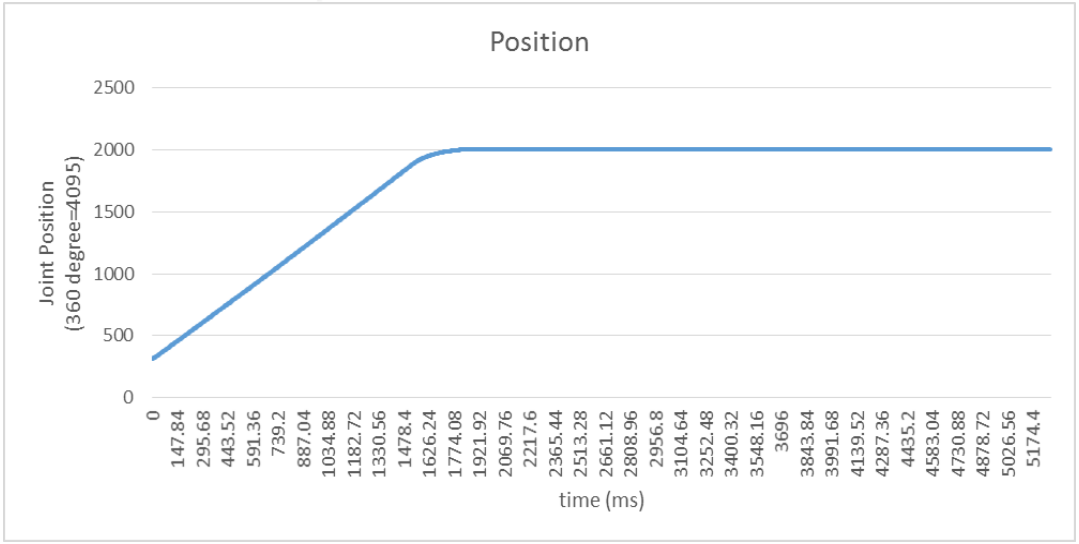


Figure 45: Joint Position When Moving Forward

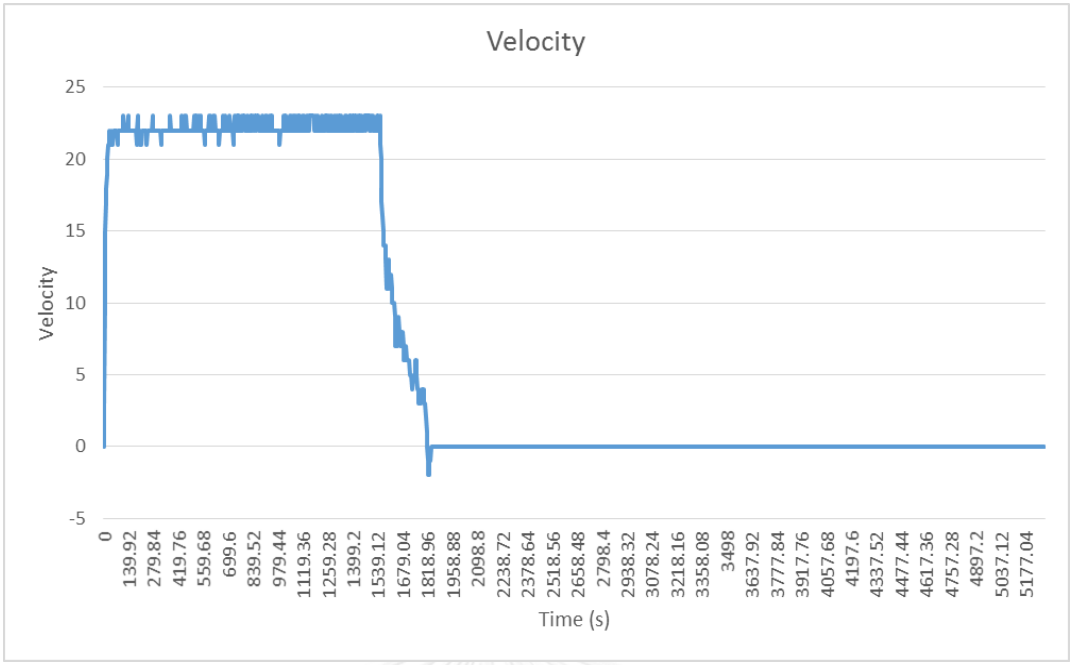


Figure 46: Motor Velocity When Moving Forward

Figure 47, Figure 48 and Figure 49 show the joint moving backward to read the desire point.

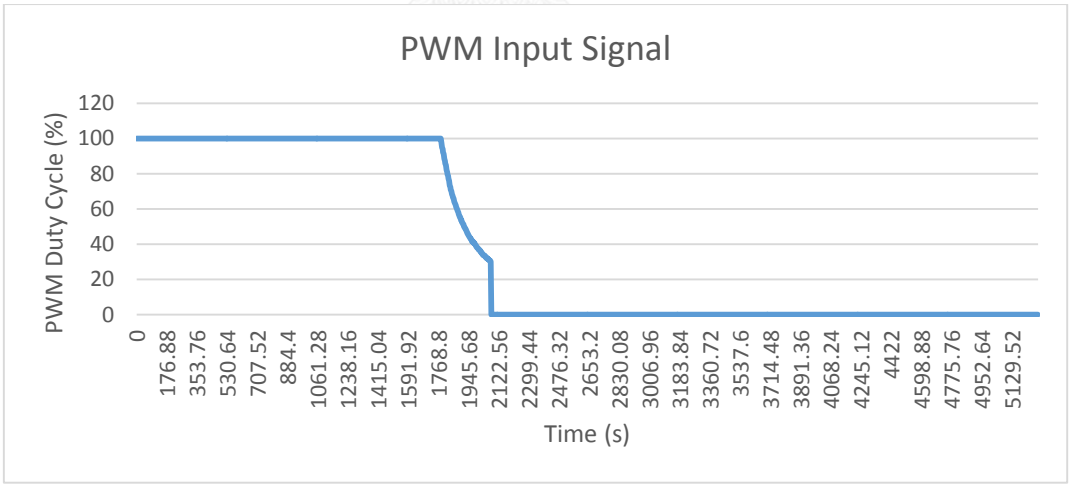


Figure 47: PWM Input Signal When Moving Backward

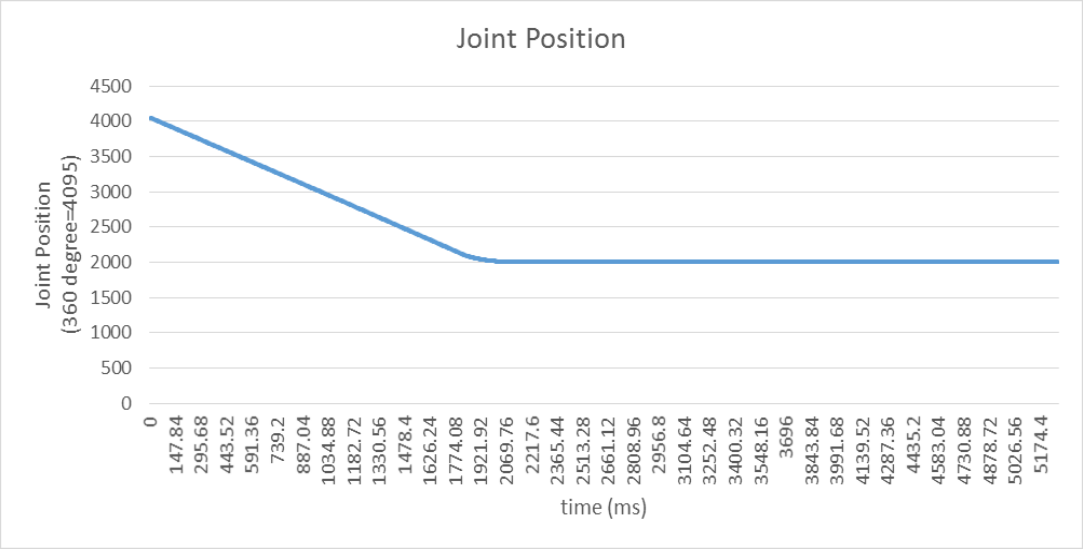


Figure 48: Joint Position When Moving Backward

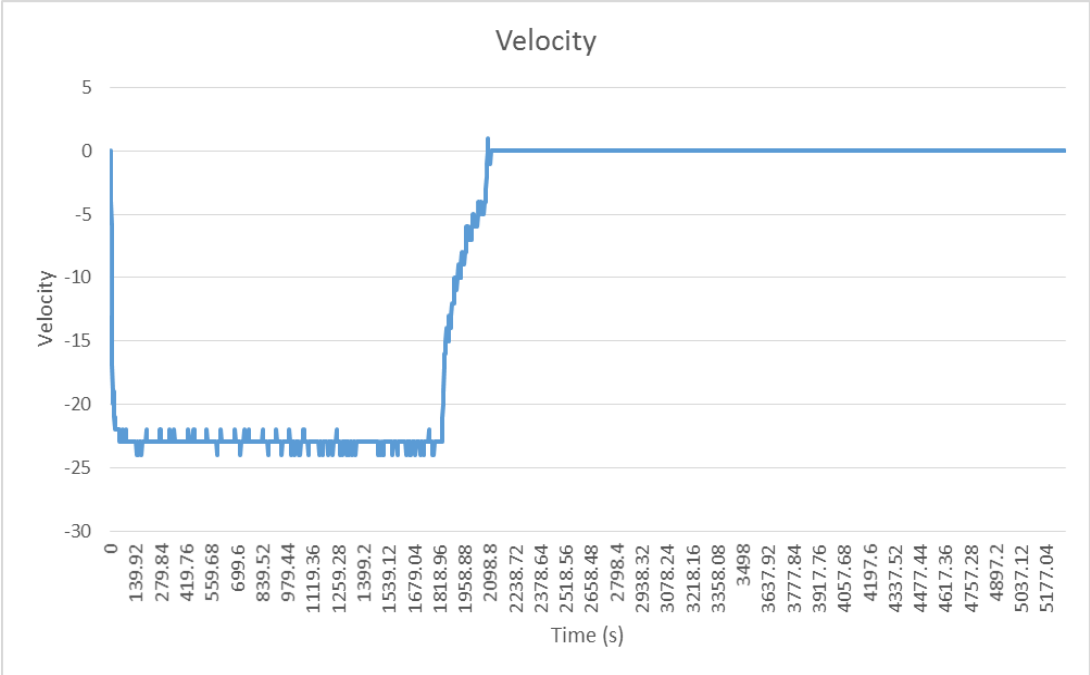


Figure 49: Motor Velocity When Moving Backward

As the figures show above for the position output they are moving smoothly toward it destination. For the velocity of the output that we get, they got some small fluctuation along the way before they reach the destination and become zero speed. This fluctuation is causes by the varied friction along the way at robot joint when it is moving.

CHAPTER 5
CONCLUSIONS

The objective of this thesis is to study the characteristic of all joints of PA10 Robot Arm in order to construct a low cost joint robot controller prototype board based on simple controller for the robot arm which can drive the joint of the Mitsubishi PA10 Robot Arm.

This research achieves its aim to build the prototype joint controller board. The accurate and high resolution position sensors (resolver) gave the exact rotor position of the motor rotor which lead us to find the flux stator needed for motor driving easier. Through the theories and hardware design in chapter 3 we can define the needed component for the controlling the robot joint, and we can make circuit schematic and PCB layout needed for creating PCB board of the controller. The technic that we combine the calculation of sine and PWM duty cycle needed for driving the motor and store it in array do help us to save the computation time of the microcontroller. We have built a graphic user inter face in MatLab with the use of serial communication RS485 as communication medium for user to send the command to the controller bard easily. Finally, we have successfully controlled the joint of Mitsubishi PA10 Robot Arm to move forward, backward to desired position according to position command sent from PC as the result has shown in the chapter 4.

Moreover to enhance this work with current completed work, in the future work, it is recommended that the protocol for the communication between the controller board and PC and the speed control should be improved when using it to control the whole robot especially in the application which require specific timing control like trajectory planning etc.

REFERENCES

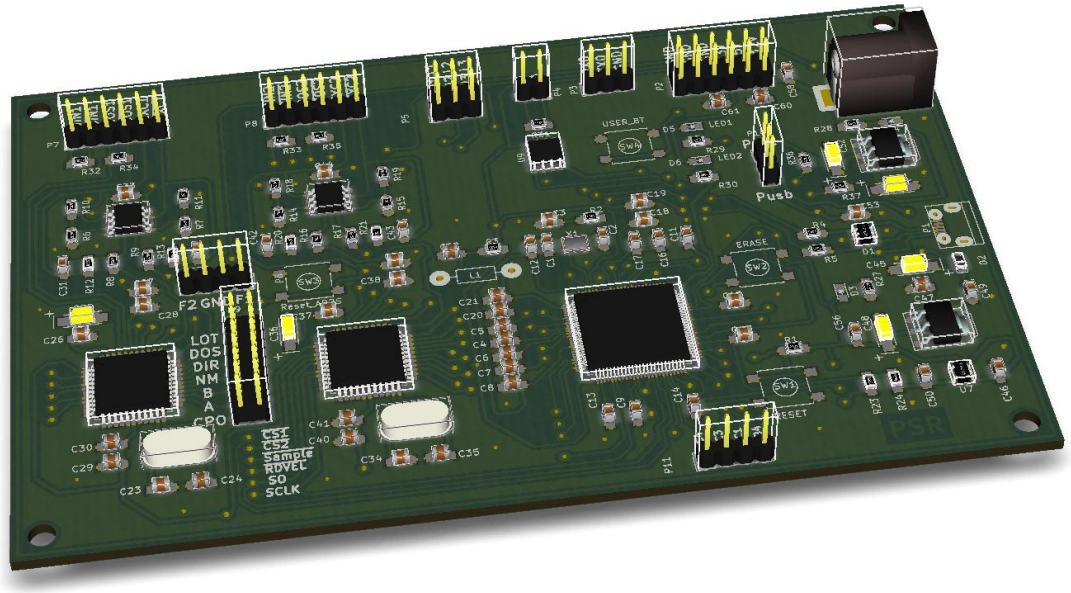
1. *Portable General Purpose Intelligent Arm Operating Manual*, L. Mitsubishi Heavy Industries, Editor.
2. Traver, J.J.S. *Robotic Intelligence Lab*. 2010 [cited 2016 03 July 2016]; Available from: http://www.robot.uji.es/lab/plone/Members/jsorribe.
3. *PMSM*. 2015 [cited 2015; Available from: http://uniquemoments.ro/.
4. Pillay, P. and R. Krishnan, *Application characteristics of permanent magnet synchronous and brushless DC motors for servo drives.* Industry Applications, IEEE Transactions on, 1991. **27**(5): p. 986-996.
5. Sachs, J. *Three-phase PWM, and zero-sequence voltage*. 2015 10 December 2015]; Available from: http://microchip.wikidot.com/mct5001:pwm.
6. *RS485 serial information*. 2015 [cited 2015 05 July 2016]; Available from: http://www.lammertbies.nl/comm/info/RS-485.html.
7. *RS485 serial information*. Available from: http://www.lammertbies.nl/comm/info/RS-485.html.
8. *AD2S1205*, I. Analog Devices, Editor. 2007.
9. Pillay, P., *Application Characteristics of Permanent Magnet Synchronous and Brushless dc Motors for Servo Drives.* IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS, 1991. **27**.
10. J. Symczak, S.O.M., J. S. Gealon, C. Nelson *Precision Resolver-to-Digital Angular Position and Velocity*. 2014.
11. *LM1117/LM1117I 800mA Low-Dropout Linear Regulator*, N. Semiconductor, Editor. 2002.
12. *SAM4S Series*, Atmel, Editor. 2015.
13. *Getting Started in KiCad*. 2016. 54.
14. Atmel. *Atmel Studio*. 2016 [cited 2016 01 July 2016]; Available from: http://www.atmel.com/tools/atmelstudio.aspx.
15. Wilson, D., *Intro to Field Oriented Control*. 2016, Texas Instuments.
16. Wilson, D., *Intro to Field Oriented Control*. 2014. p. www.kappaiq.com.
17. Eskander, M.N., O.M. Arafa, and O.A. Mahgoub. *Sensorless Control of PMSM and BDCM Based On EMF Extraction And Extended Kalman Estimator*. in *2006 IEEE International Symposium on Industrial Electronics*. 2006.
18. Wang, Z., L. Quan, and X. Liu, *Sensorless SPMSM Position Estimation Using Position Estimation Error Suppression Control and EKF in Wide Speed Range.* Mathematical Problems in Engineering, 2014. **2014**: p. 11.
19. Atmel. *Atmel SAM-BA In-system Programmer*. 2016 [cited 2016 29-06-2016]; Available from: http://www.atmel.com/tools/atmelsam-bain-systemprogrammer.aspx.

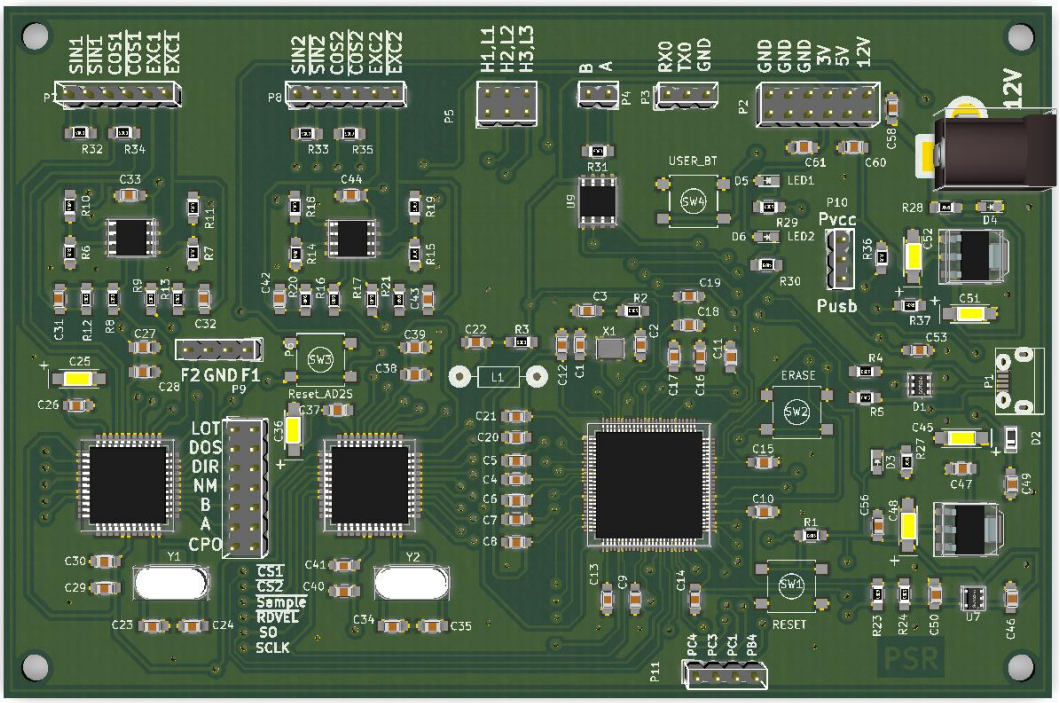
APPENDIX

List of Publication

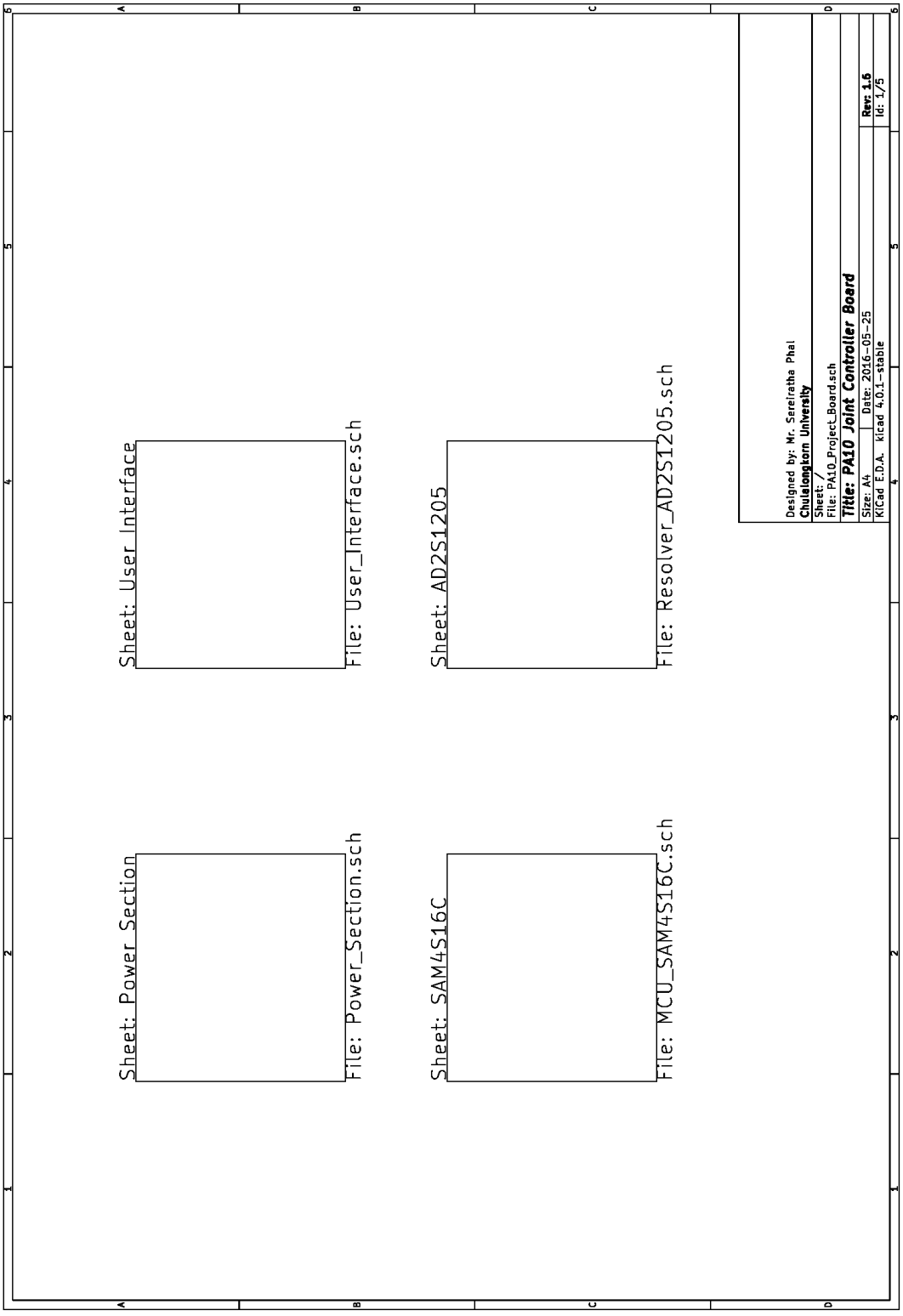
Sereiratha Phal and Manop Wongsaisuwan, “Design of Joint Controller Circuit for PA10 Robot Arm,” *in the 4th International Conference on Engineering & ICT 2016 (ICEI 2016)*, Melaka, Malaysia, and published in Journal of Telecommunication, Electronic and Computer Engineering (JTEC).

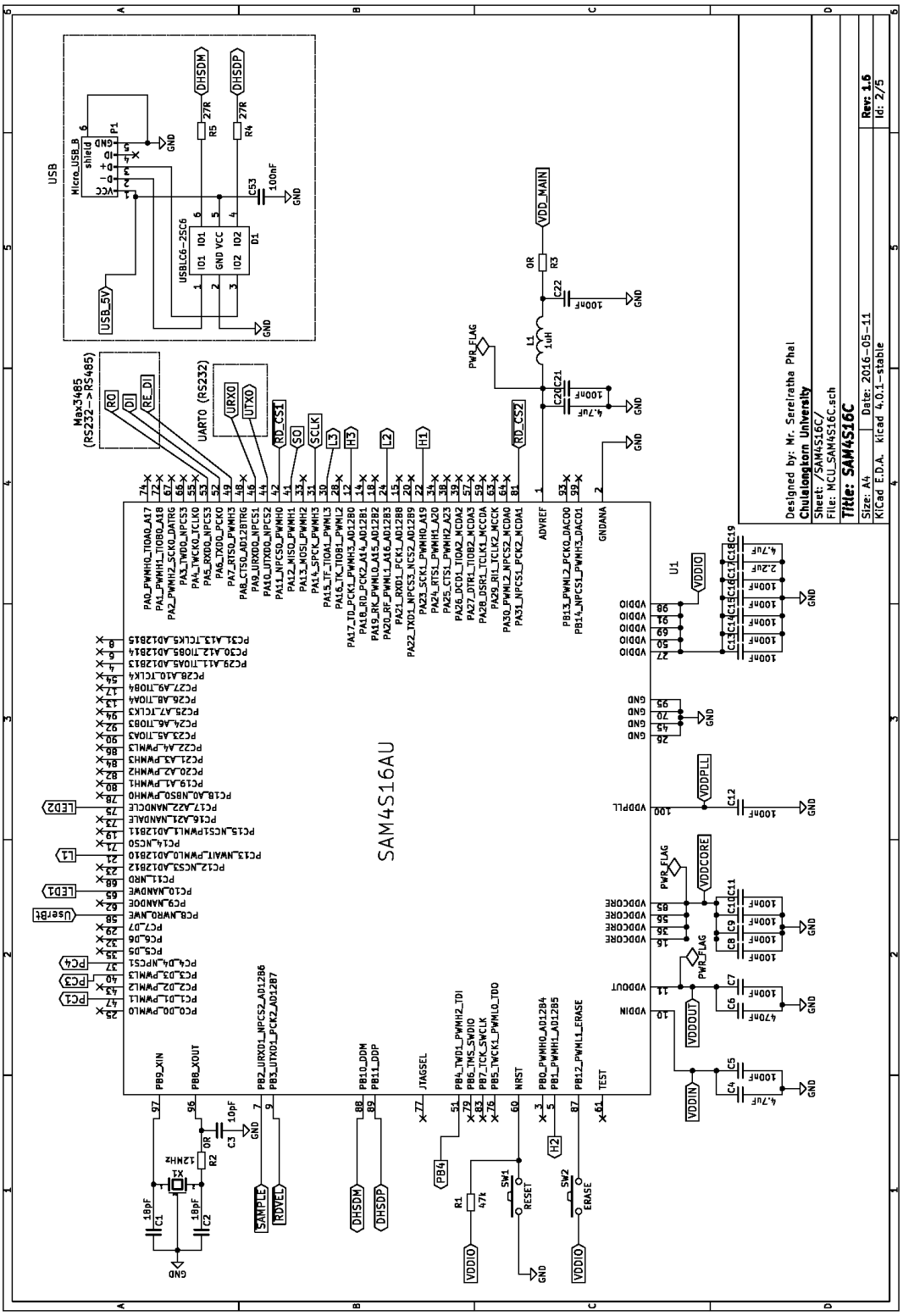
Controller Board in 3D

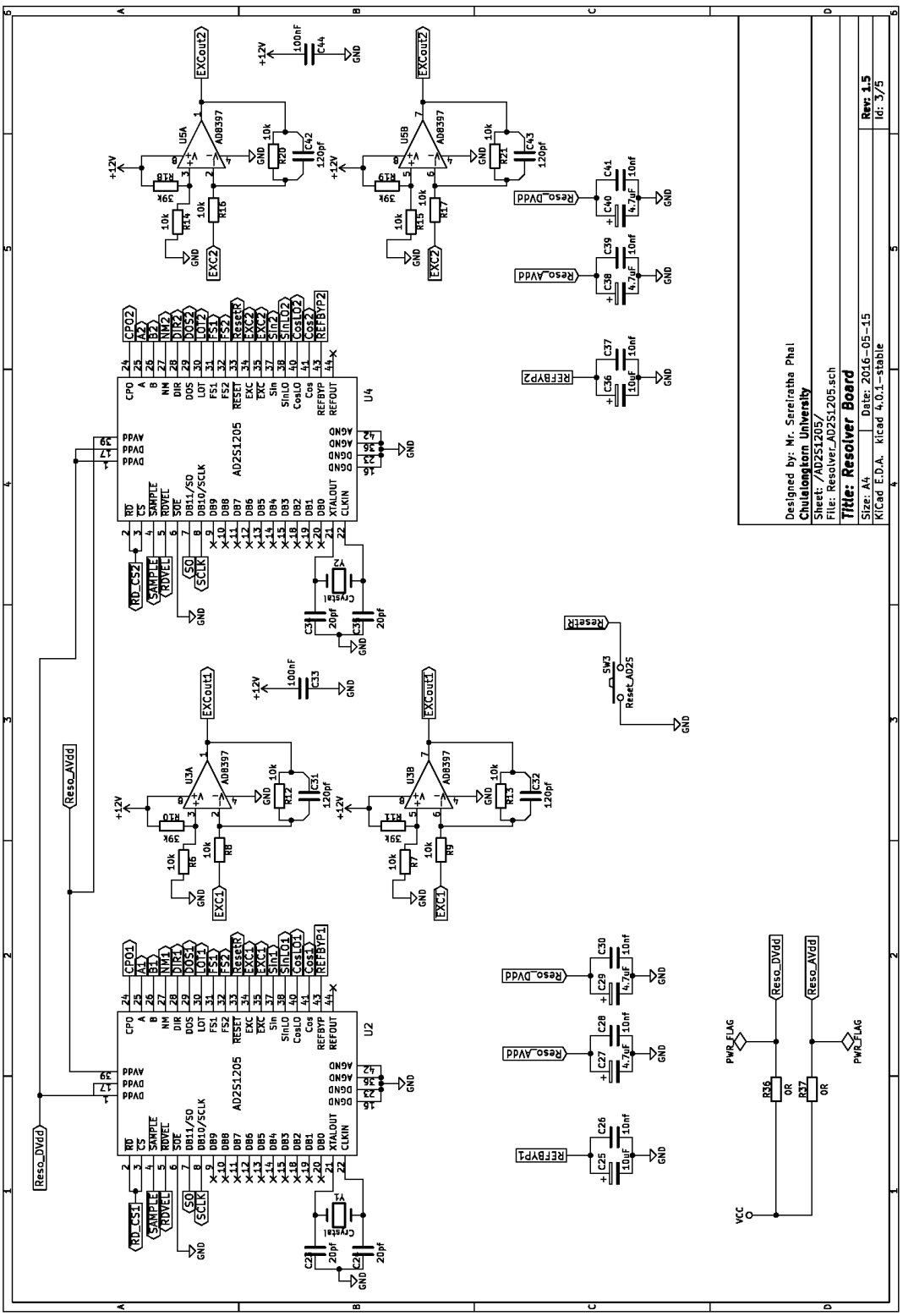


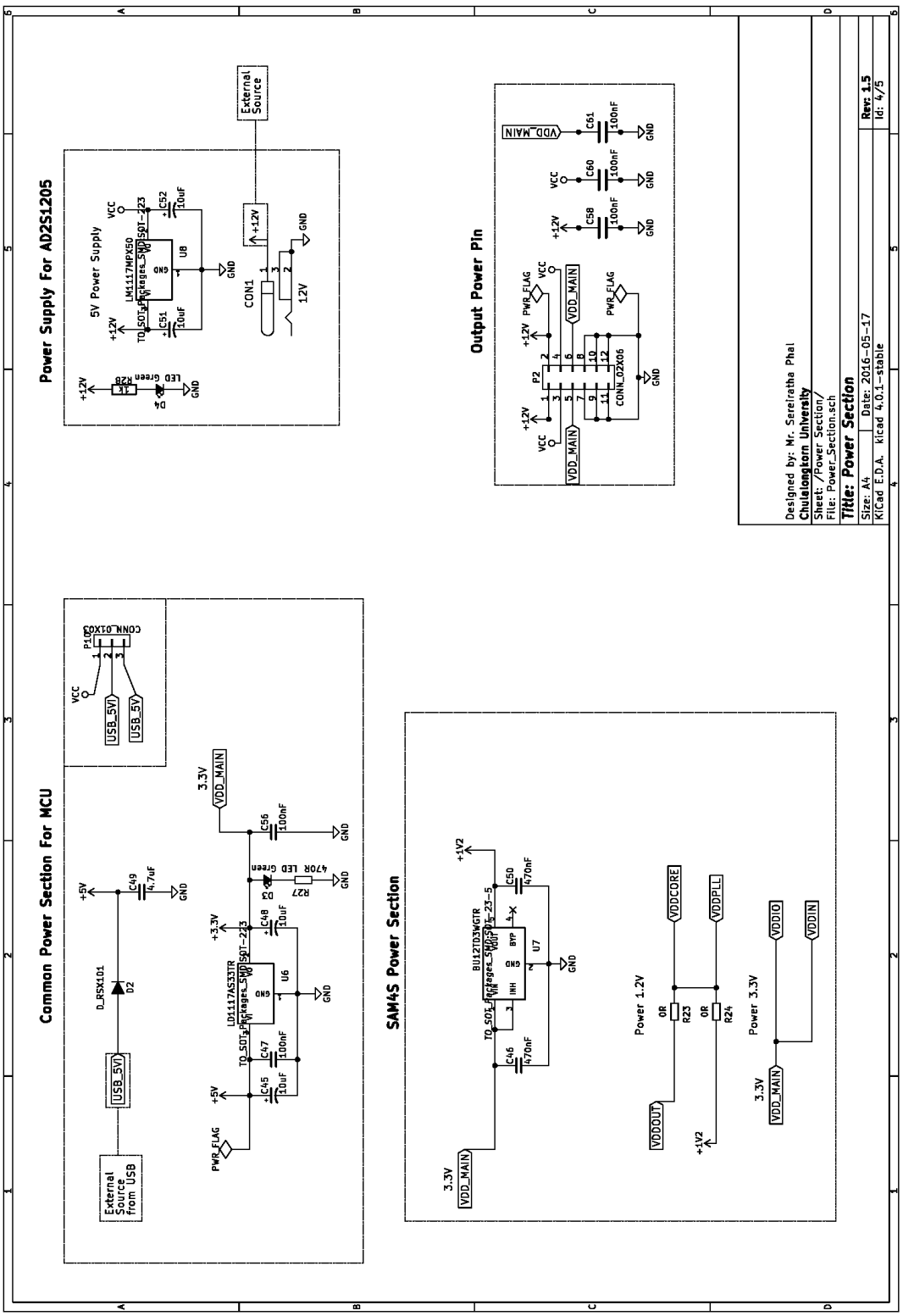


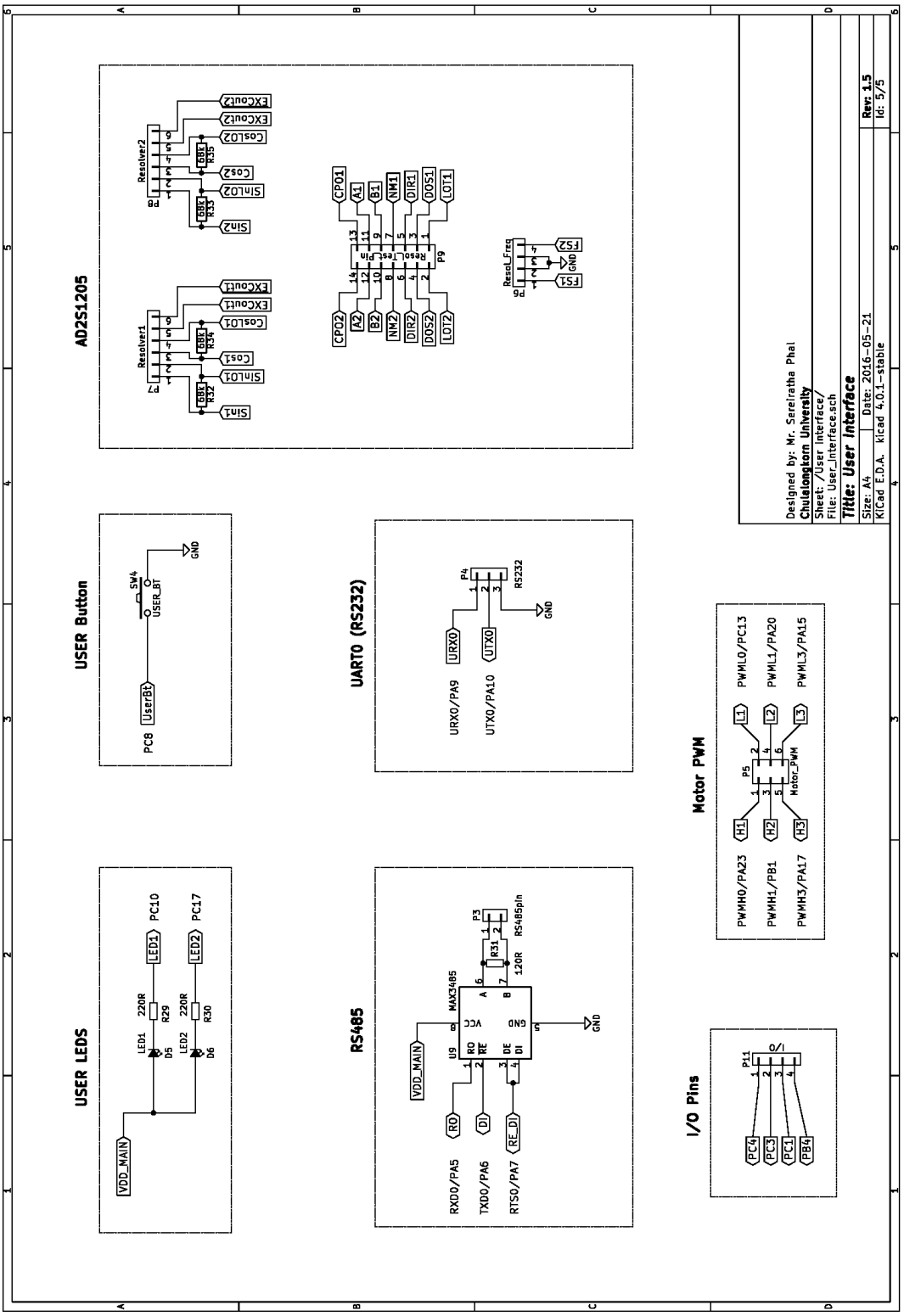
Controller Schematic




SAM4S16AU
USB
Micro_USB_B
shield
P1
GND
VCC
D-
D+
ID
USB_5V
USBLC6-2SC6
IO1 IO1
GND VCC
IO2 IO2
D1
27R
R5
R4
DHSDM
DHSDP
C53
100nF
Max3485
(RS232->RS485)
RO
DI
RE_DI
UART0 (RS232)
URX0
UTX0
RD_CS1
SO
SCLK
L3
H3
L2
H1
RD_CS2
VDD_MAIN
0R
R3
C22
100nF
L1
1uH
PWR_FLAG
C20
4.7uF
C21
100nF
ADVREF
GNDANA
U1
VDDIO
VDDPLL
VDDCORE
VDDOUT
VDDIN
C12
C13
C14
C15
C16
C17
C18
C19
2.2uF
4.7uF
100nF
C9
C10
C11
C7
C6
C5
C4
C3
10pF
R2
0R
12MHz
X1
C1
C2
18pF
SAMPLE
ROVEL
R1
47k
SW1
RESET
SW2
ERASE
H2
PB4
JTAGSEL
NRST
TEST
PB10_DDM
PB11_DDP
PB8_XOUT
PB9_XIN
PB2_URXD1_NPCS2_AD1286
PB3_UTXD1_PCK2_AD1287
PB4_TWD1_PWMH2_TDI
PB6_TMS_SWDIO
PB7_TCK_SWCLK
PB5_TWCK1_PWML0_TDO
PB0_PWMH0_AD1284
PB1_PWMH1_AD1285
PB12_PWML1_ERASE
PB13_PWML2_PCK0_DAC0
PB14_NPCS1_PWMH3_DAC1
LED2
L1
LED1
UserBt
PC4
PC3
PC1
Designed by: Mr. Sereiratha Phal
Chulalongkorn University
Sheet: /SAM4S16C/
File: MCU_SAM4S16C.sch
Title: SAM4S16C
Size: A4
Date: 2016-05-11
KiCad E.D.A. kicad 4.0.1-stable
Rev: 1.6
Id: 2/5


Designed by: Mr. Sereiratha Phal
Chulalongkorn University
Sheet: /AD2S1205/
File: Resolver_AD2S1205.sch
Title: Resolver Board
Size: A4
Date: 2016-05-15
Rev: 1.5
KiCad E.D.A. kicad 4.0.1-stable
Id: 3/5
AD2S1205
AD8397
Reso_AVdd
Reso_DVdd
RD_CS1
RD_CS2
SAMPLE
RDVEL
SDI
SCLK
EXCout1
EXCout2
EXC1
EXC2
Reset_AD2S
PWR_FLAG
VCC


Power Supply For AD2S1205
5V Power Supply
LM1117MPX50
SOT-223
LED Green
External Source
CON1
1.2V
Output Power Pin
VDD_MAIN
PWR_FLAG
CONN_02X06
Common Power Section For MCU
CONN_01X03
USB_5V
External Source from USB
D_ASK101
LD1117AS33TR
470R LED Green
SAM4S Power Section
BU12TD3WGTR
Power 1.2V
Power 3.3V
VDDCORE
VDDPLL
VDDIO
VDDIN
VDDOUT
Designed by: Mr. Sereiratha Phal
Chulalongkorn University
Sheet: /Power Section/
File: Power_Section.sch
Title: Power Section
Size: A4
Date: 2016-05-17
Rev: 1.5
KiCad E.D.A. kicad 4.0.1-stable
Id: 4/5


AD2S1205
Resolver1
Resolver2
ExCout1
ExCout2
CosLO1
CosLO2
Cos1
Cos2
SinLO1
SinLO2
Sin1
Sin2
R32
R33
R34
R35
P7
P8
P9
CPO1
CPO2
A1
A2
B1
B2
NM1
NM2
DIR1
DIR2
DOS1
DOS2
LOT1
LOT2
Test
Reso
ResolFreq
FS1
FS2
GND
USER Button
SW4
USER_BT
UserBt
PC8
UART0 (RS232)
P4
RS232
URX0
UTX0
URX0/PA9
UTX0/PA10
RS485
P3
RS485pin
R31
120R
MAX3485
VCC
GND
RO
RE
DE
DI
U9
VDD_MAIN
RXD0/PA5
TXD0/PA6
RTS0/PA7
RE_DI
USER LEDS
LED1
LED2
220R
R29
R30
D5
D6
PC10
PC17
Motor PWM
P5
Motor_PWM
H1
H2
H3
L1
L2
L3
PWMH0/PA23
PWMH1/PB1
PWMH3/PA17
PWML0/PC13
PWML1/PA20
PWML3/PA15
I/O Pins
P11
PC4
PC3
PC1
PB4
Designed by: Mr. Sereiratha Phal
Chulalongkorn University
Sheet: /User Interface/
File: User_Interface.sch
Title: User Interface
Size: A4
Date: 2016-05-21
KiCad E.D.A. kicad 4.0.1-stable
Rev: 1.5
Id: 5/5

Component List

PA10 JOINT CONTROLLER BOARD BOM REV 1.6

DATE	COMPANY	COMMENT 1	COMMENT 2
2016-05-25	Chulalongkorn University	Designed by: Mr. Sereiratha Phal	

COMMENT 3	COMMENT 4	TOTAL PARTS
		126 (Unique 51)

Ref	Qty	Value	Footprint
[C] - Capacitors			
C13 C14 C15 C16 C17 C12 C8 C9 C10 C11 C5 C7 C21 C22 C53 C44 C33 C56 C58 C60 C61 C47	22	100nF	Capacitors_SMD:C_0805_HandSoldering
C18	1	2.2uF	Capacitors_SMD:C_0805_HandSoldering
C19 C4 C20 C27 C29 C38 C40 C49	8	4.7uF	Capacitors_SMD:C_0805_HandSoldering
C6 C46 C50	3	470nF	Capacitors_SMD:C_0805_HandSoldering
C2 C1	2	18pF	Capacitors_SMD:C_0805_HandSoldering
C3	1	10pF	Capacitors_SMD:C_0805_HandSoldering
C32 C31 C43 C42	4	120pf	Capacitors_SMD:C_0805_HandSoldering
C26 C28 C30 C37 C39 C41	6	10nf	Capacitors_SMD:C_0805_HandSoldering
C25 C36 C45 C48 C51 C52	6	10uF	Capacitors_Tantalum_SMD:TantalC_SizeA_EIA-3216_HandSoldering
C23 C24 C34 C35	4	20pf	Capacitors_SMD:C_0805_HandSoldering
[CON] -			
CON1	1	12V	Connect:JACK_ALIM
[D] - Diodes			
D1	1	USBLC6-2SC6	TO_SOT_Packages_SMD:SOT-23-6
D3 D4	2	LED Green	LEDs:LED_0805
D2	1	D_RSX101	Diodes_SMD:TUMD2
D5	1	LED1	LEDs:LED_0805
D6	1	LED2	LEDs:LED_0805
[L] - Inductors			
L1	1	1uH	Choke_Axial_ThroughHole:Choke_Horizontal_RM10mm
[P] - Connectors			
P1	1	Micro_USB_B	Connect:USB_Micro-B
P2	1	CONN_02X06	Pin_Headers:Pin_Header_Straight_2x06
P10	1	CONN_01X03	Pin_Headers:Pin_Header_Straight_1x03
P5	1	Motor_PWM	Pin_Headers:Pin_Header_Straight_2x03
P4	1	RS232	Pin_Headers:Pin_Header_Straight_1x03
P3	1	RS485pin	Pin_Headers:Pin_Header_Straight_1x02
P7	1	Resolver1	Pin_Headers:Pin_Header_Straight_1x06
P8	1	Resolver2	Pin_Headers:Pin_Header_Straight_1x06
P6	1	Resol_Freq	Pin_Headers:Pin_Header_Straight_1x04
P9	1	Resol_Test_Pin	Pin_Headers:Pin_Header_Straight_2x07
P11	1	I/O	Pin_Headers:Pin_Header_Straight_1x04

[R] - Resistors			
R2 R3 R37 R36 R23 R24	6	0R	Resistors_SMD:R_0805_HandSoldering
R1	1	47k	Resistors_SMD:R_0805_HandSoldering
R4 R5	2	27R	Resistors_SMD:R_0805_HandSoldering
R13 R9 R7 R12 R8 R6 R21 R17 R15 R20 R16 R14	12	10k	Resistors_SMD:R_0805_HandSoldering
R11 R10 R19 R18	4	39k	Resistors_SMD:R_0805_HandSoldering
R27	1	470R	Resistors_SMD:R_0805_HandSoldering
R28	1	1k	Resistors_SMD:R_0805_HandSoldering
R29 R30	2	220R	Resistors_SMD:R_0805_HandSoldering
R31	1	120R	Resistors_SMD:R_0805_HandSoldering
R34 R32 R35 R33	4	68k	Resistors_SMD:R_0805_HandSoldering
[SW] -			
SW2	1	ERASE	Buttons_Switches_SMD:SW_SPST_B3S-1000
SW1	1	RESET	Buttons_Switches_SMD:SW_SPST_B3S-1000
SW3	1	Reset_AD2S	Buttons_Switches_ThroughHole:SW_PUSH_SMALL
SW4	1	USER_BT	Buttons_Switches_SMD:SW_SPST_B3S-1000
[U] - Semiconductors			
U1	1	SAM4S16AU	Housings_QFP:LQFP-100_14x14mm_Pitch0.5mm
U2 U4	2	AD2S1205	Housings_QFP:LQFP-44_10x10mm_Pitch0.8mm
U3 U5	2	AD8397	Housings_SOIC:SOIC-8_3.9x4.9mm_Pitch1.27mm
U8	1	LM1117MPX50	TO_SOT_Packages_SMD:SOT-223
U6	1	LD1117AS33TR	TO_SOT_Packages_SMD:SOT-223
U7	1	BU12TD3WGTR	TO_SOT_Packages_SMD:SOT-23-5
U9	1	MAX3485	Housings_SOIC:SOIC-8_3.9x4.9mm_Pitch1.27mm
[X] - Crystals			
X1	1	12MHz	Crystals:crystal_FA238-TSX3225
[Y] -			
Y1 Y2	2	Crystal	Crystals:HC-49V

BOM made with KICAD_BOM_WIZARD

VITA

Mr. Sereiratha Phal was born in 1990, in Kompong Chhnang Province, Cambodia. He was rewarded Samdech Hun Sen -Handa National Scholarships 2008 in October 2008 to study in Bachelor's Degree Program majoring in Information Technology at University of Cambodia and graduated in September 2012. He also got an Engineer's Degree in Electrical and Electronic Engineering from Institute of Technology of Cambodia in July 2013. In October 2013, he was rewarded AUN/SEED-Net scholarships for the Master's Degree Program to study at Chulalongkorn University. His research interest are information technology (IT), embedded systems and robotics.