INSERTING THE CELLS OF BATTERY USING ROBOTIC ARM

By

NC-SHEHARYAR

NC-NAVEED RAMZAN

NC-HAFIZ MUHAMMAD USMAN

PC-MOIN-UL-HAQ

Submitted to the Faculty of Electrical Engineering, Military College of Signals, National University of Science and Technology, Rawalpindi in partial fulfillment for the requirements of a B.E. Degree in Electrical (Telecom) Engineering

JUNE 2014

CERTIFICATE OF CORRECTNESS AND APPROVAL

It is certified that the work contained in this thesis "INSERTING THE CELLS OF BATTERY USING ROBOTIC ARM", was carried out by Sheheryar, Naveed Ramzan, Hafiz Muhammad Usman and Moin Ul Haq under the supervision of Asst Prof Imtiaz Ahmed Khokhar for partial fulfillment of Degree of Bachelor of Telecommunication Engineering, is correct and approved.

Approved by

_____

Asst Prof Imtiaz Ahmed Khokhar

EE Department

MCS

Dated: _____

ABSTRACT

Cells of battery are made from combination of lead plates covered with Absorbent Glass Mat (AGM). Traditionally cell insertion in the battery container is done by hand, AGM is a soft mat and cell is always very tight in its casing so generally, mat gets folded while inserting the cells into the battery container causing variations in pressure on the plates and in turn decreases the efficiency as well as life time of the battery. The goal of this project is to make automation for the cell insertion process. The automation would include designing of a robotic arm and conveyor belt using sliding mode controller, robotic arm will take cells one by one from the rack and insert them into the battery container without causing the AGM to fold and damage the cells thus increasing the efficiency.

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

In the name of Allah, the Most Gracious, the Most Beneficent,

Dedicated to our parents, our teachers and especially Military College of Signals.

ACKNOWLEDGEMENTS

There is no success without the will of ALLAH Almighty. We are obliged to ALLAH, who has given us guidance, power and enabled us to achieve this task. Whatever we have achieved, we owe it to Him, in totality. We are thankful to our parents who supported us both morally and financially. We thank our friends for their admirable support and their critical reviews.

Due extension of appreciation to our project supervisor Asst Prof Imtiaz Ahmed Khokhar for his continuous technical supervision and moral support during the project. His calm nature ensured smooth work and his composed manner guided us work around obstacles with determination.

We would like to extend Special thanks to Lt Col Imran Rashid for his guidance throughout the completion of our project.

Table of Contents

Table of Figures

List of Tables

List of Abbreviation

AGM	Absorbent Glass Mat
A.C	Alternating Current
CW	Clockwise
CCW	Counter Clockwise
D.C	Direct Current
IR	Infrared
LED	Light Emitting Diode
PID	Proportional, Integral, Derivative
PVC	Polyvinylchloride
SRAM	Static random-access memory
SCM	Sliding mode controller
EEP	Electrically Erasable Programmable
I/O	Input/output
RPM	Revolution per Minute
Nm	Newton Meter
USB	Universal Serial Bus
ICSP	In-Circuit Serial Programming

CHAPTER ONE

INTRODUCTION

CHAPTER 1

1.1 Introduction

Robot is an essential part in automating the flexible developing system that one significantly demand these days. Robots are now above a engine, as robots have become the key of the future as cost labor salary and clients order. Even if the cost of establishing robotic system is relatively costly but as today's swift growth and a high challenging superiority with IS0 (International Standard Organization), human are no longer skilled of such demands. Research and development of future robots is moving at a very fast rate due to the continuously improving and improving of the quality standards of goods. Robot and automation is engaged in order to change human to carry out those responsibilities that are regular, unsafe, and boring and in a unsafe area. In a world of complex technology today, Automation significantly increases production ability. Improve product quality and lower manufacturing cost. It takes just a few people to plan or check the computer and perform routine preservation.

It is desirable in many applications to have robot arms which can follow a specific trajectory. If a multi-joint robot arm tries to follow a desired trajectory, the appropriate motor commands must be applied to each joint of the arm at all times during the motion. Every joint is dependent on its parent joints, and wrong motor commands can lead to undesirable movements; this can result in unpredictable consequences which may have an adverse impact on the robot's task and its environment. Furthermore, the correct motor commands must be adapted over time in order to take into consideration changing physical parameters such as arm load, orientation, or joint friction.

Automated robotic arms have matured greatly over the last 40 years. These systems are now implemented in the industries all over the world to maximize the efficiency.

1.2 Background

During the past few years automation is done in industries. In many areas robots are implemented in assembly lines for increasing the quality as well as efficiency and also to reduce the production cost. The most important question is what to automate. We have to analyze that at what places human are better and at what place machines are powerful. Instead the most important tactic today is that how both robots and human work together

.Several projects have been developed to increase the cooperation between human and machines. It is very important to have a good communication between workers whether they are machines or human.

In battery manufacturing industries traditional way of inserting the cells of the battery in the casing of the battery is through hands. Size of the battery cells and the container is approximately the same due to which only chance for inserting the cell through hand is to pick cell from the top, while doing this A.G.M (Absorbent Glass Mat) of the cell get folded and puts pressure on the plates of the battery and in turn decreases the efficiency and life time of the battery.

Figure 1 : Absorbent Glass Mat

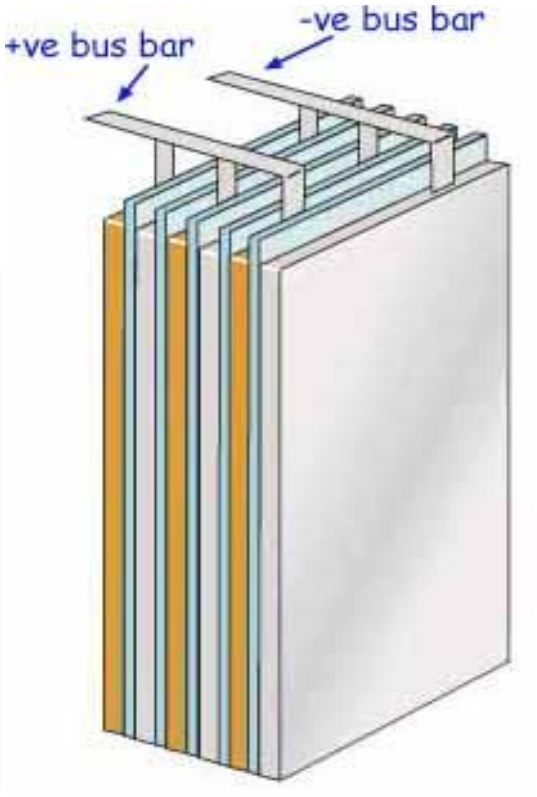

Figure 2 : Battery Cell

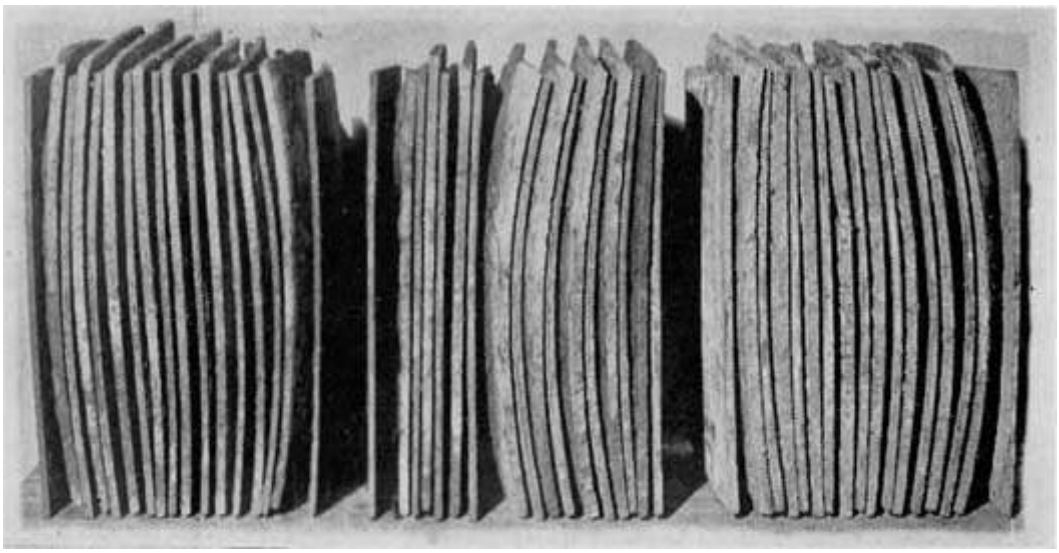

Figure 3 : Folded Battery Cells

1.3 Objectives

- Programming Arduino for real time communication between arduino and SCILAB.
- Development of mechanical structure of conveyor belt which control the position of the battery casing and the battery cells.
- Development of mechanical structure of robotic arm to pick the cells from the conveyor belt and place them in the case of the battery
- Development of SCILAB model to control the movements of robotic arm and conveyor belts.
- Implementing (SMC) sliding mode controller and PID controller.
- Interfacing robotic arm and conveyor belts with the SCILAB software.

1.4 Outline of Tasks

Firstly research was carried out to determine how to communicate between computer (SCILAB) and the external world. Different DSP kits and boards were studied in this research. Then we have to design mechanical structure of conveyor belts and robotic arm in software which will be suitable for our project. Then we have developed SCILAB model for controlling hardware. Finally we have interface our mechanical structure and software and develop prototype of Robotic arm.

CHAPTER 2
LITERATURE REVIEW

LITERATURE REVIEW

2.1 Overview:

Robotic arm is an essential element in automating the manufacturing method which is really in demand these days. Robotic arms are now above machines, as these are engaged in order to replace humans to carry out those responsibilities that are regular, unsafe, and boring or are in a hazardous area. In a world of complex technology today, automation significantly increases production facility; develop product feature and lesser production cost. It takes just a few people to plan or supervise the computer and perform schedule preservation.

2.2 Control Theory

Control theory is an interdisciplinary division of engineering and mathematics that deals with the performance of dynamical systems. The preferred output of a system is called the reference. When one or more output variables of a system need to follow a definite reference ultimately, a controller manipulates the inputs to a system to achieve the preferred result on the output of the system.

The typical goal of a control theory is to determine solutions for the suitable corrective action from the controller that result in system constancy, states that system will hold the set point and not fluctuate around it.

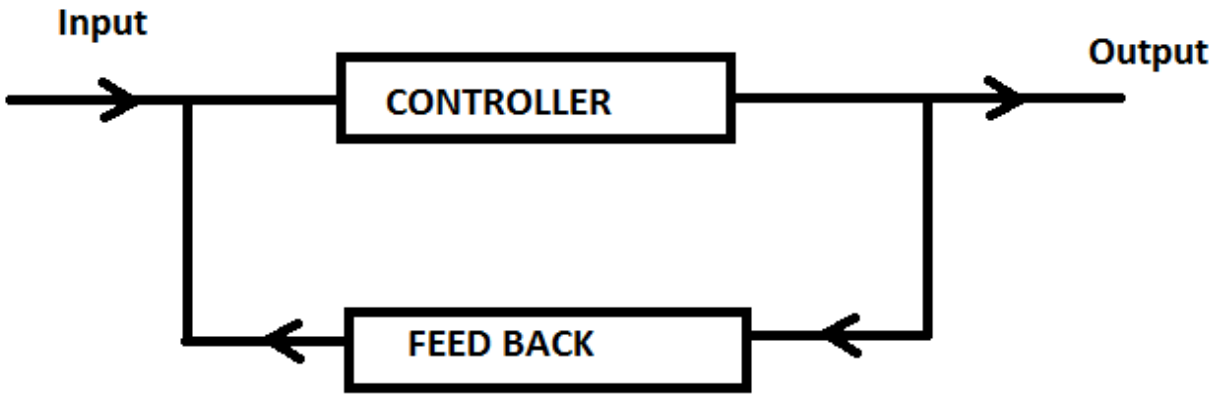

Figure 4: Feedback Loop to Control the Dynamic Behavior of the Reference

2.3 PID Controller

PID Control (proportional-integral-derivative) is certainly the widest type of regular control used in manufacturing. Although it has a moderately simple algorithm/composition, there are many appropriate variations in how it is useful in production. A proportional–integral–derivative controller (PID controller) is a standard control loop feedback method commonly used in industrialized control systems. A PID controller will correct the error between the output and the preferred input or set point by manipulating and provide an output for improvement that will regulate the method consequently. A PID controller has the general form.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{de}{dt}$$

Where Kp is proportional gain, Ki is the integral gain, and Kd is the derivative gain. The PID controller computation involves three separate parameters; the Proportional, the Integral and Derivative principles. The Proportional value determines the response to the present error, the Integral determines the response based on the sum of recent errors and the Derivative determines the response to the rate at which the error has been varying. The weighted sum of these three events is used to correct the procedure using a control part for instance the position of a control valve, the power supply of a heating element or DC motor speed and position.

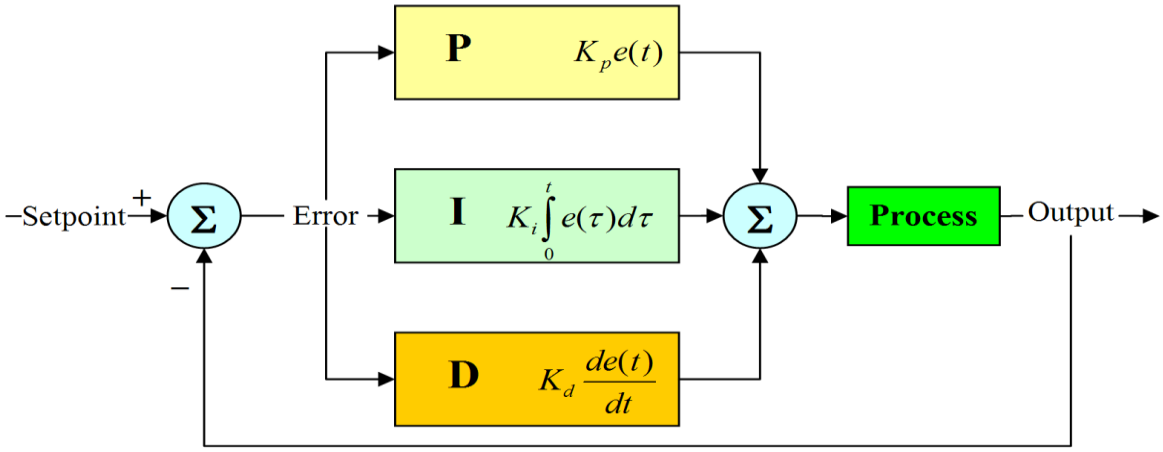

Figure 5 : PID

2.4 Sliding mode controller

Sliding mode control is an essential robust control approach and has smart features to maintain the systems insensible to the suspicions on the sliding surface. The major drawback of sliding mode control is small sensitivity to variations and interruption which eliminates the necessity of precise modeling. Sliding mode control implies that control events are irregular states functions which can simply be employed by usual power convertors with ON OFF as the only acceptable condition, sliding mode controller has proved to be appropriate to extensive variety of problems in robotics ,electric drives and generators, procedure control, vehicle and motion control. PID control is inadequate, as it relay on model, so is responsive to real world constraint variations and disorder such as motor and load inertia variations. On the other side, sliding mode algorithms have a much simpler control composition. Unlike other sophisticated servo algorithms sliding mode control composition is easier. Its performance and simplicity of use are analogous to that of aggressive technologies.

Another benefit of sliding mode control is that the bulk of applications need only control one factor to attain best performance making it easier to utilize. The sliding mode gain K_s is the main tuning factor and is associated to system inertia: If inertia is rising, K_s should also raise. Though, additional parameters are accessible to fine tune even the toughest applications.

2.5 Arduino (MEGA-2560)

Arduino is a single-board microcontroller, proposed to compose the application of interactive items or environments more available. The hardware comprises of an open source hardware board planned around an 8-bit Atmel AVR microcontroller or a 32-bit Atmel ARM. Arduino can sense the situation by getting input from a array of sensors and can change its environment by controlling motors and other actuators. The microcontroller on the board is planned using the Arduino encoding language and the Arduino development atmosphere. Arduino projects can be stand alone or they can correspond with software running on a computer (e.g. Scilab). The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital

Input/Output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything required to sustain the microcontroller; simply attach it to a computer with a USB cable or power it with an AC to DC converter or battery to start. The Mega is well suited with most shields intended for the Arduino Duemilanove. The Mega 2560 is an update to the Arduino Mega, which it replaces.

2.6 Types of motor:

Several different motors are accessible in the market like stepper, servo, dc motors with and without gears. These different motors are used according to their applications and necessities for e.g. If we want high torque and accurate position we need to use servo motors, if we want to only position and if high torques are not mandatory then stepper motors are used .DC geared motors are used where we need high torque .For only smooth motion DC motors are used.

2.6.1 DC-Motor

DC motors appear in a range of shapes and sized even if most are cylindrical. They characteristic an output shaft which rotates at high speeds usually in the 5 000 to 10 000 rpm range. Although DC motors rotate very rapidly in common, most are not *strong* (low torque). In order to reduce the speed and amplify the torque, a gear can be used. DC motors can function in clockwise (CW) and counter clockwise (CCW) rotation.

2.6.2 DC Gear Motor

Geared DC motors can be described as an expansion of DC motor. A geared DC Motor has a gear assembly connected to the motor. The speed of motor is measured in terms of rotations of the shaft per minute and is known as RPM .The gear assembly helps in increasing the torque and decreasing the speed. Using the exact arrangement of gears in a gear motor, its speed can be decreased to any wanted number. This theory where gears decrease the speed of the vehicle but boost its torque is defined as gear reduction.

A gear motor can be both an AC (alternating current) and a DC (direct current) electric motor. Most gear motors have an output of between about 1,200 to 3,600 revolutions per minute (RPMs). These types of motors also have two different speed conditions: normal speed and the stall speed torque condition. Gear motors are mostly used to decrease speed in a series of gears, which in turn creates additional torque. This is achieved by an included series of gears or a gear box being attached to the main motor rotor and shaft via a second lessening shaft. The second shaft is then associated to the series of gears or gearbox to produce what is known as a series of reduction gears. In general words, the longer the train of reduction gears, and the lesser the output of the end, gear will be.

2.7 Infrared Sensors

An infrared sensor is an electronic device that is used to sense definite individuality of its environment by either emitting or detecting infrared radiation. It is also able of measuring heat of an item and detecting movement. Infrared waves are not visible to the human eye. Infrared radiation is the region having wavelengths longer than visible light wavelengths in the electromagnetic spectrum, but lesser than microwaves.

IR Sensors work by using a precise light sensor to sense a unique light wavelength in the Infra Red (IR) spectrum. By using an LED which produces light at the same wavelength as what the sensor is looking for, you can come across at the amount of the received light. When an object is close to the sensor, the light from the LED bounces off the object and into the light sensor. This results in a larger in the intensity, which we know can be detected using a limit.

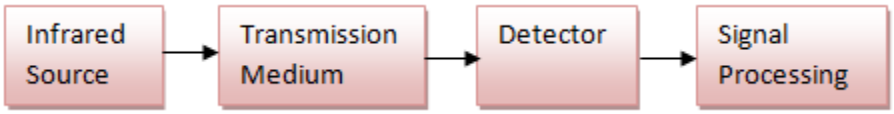

Figure 6 : Infrared Mechanism

2.7.1 Infrared Source

All objects above 0 K emit infrared energy and therefore are infrared sources. Infrared sources also consist of blackbody radiators, tungsten lamps, silicon carbide, and a variety

of others items. For active IR sensors, infrared Lasers and LEDs of definite IR wavelengths are used as IR sources.

2.7.2 Transmission Medium

Three major type of transmission medium used for Infrared transmission are vacuum, the atmosphere, and optical fibers.

2.7.3 Infrared detectors

Different types of detectors are used in IR sensors. Most important condition of detector is Photosensitivity. It is the Output Voltage/Current per watt of incident energy. Higher value is better for future use.

2.7.4 Signal Processing

As detector outputs are usually very small, preamplifiers with related circuitry are used to further procedure the received signals.

2.8 Absorbent Glass Mat (AGM)

In early 1985 AGM technology became accepted. It was used as sealed lead acid battery for military aircraft, vehicles and UPS for the function to decrease weight and to recover consistency. Fine fiberglass mat absorbs the acid and build battery spill proof. Hazardous material limitations were removed in its shipment. Its plates are made flat to look like standard flooded lead acid pack in a rectangular case.

AGM has low internal resistance, is able of delivering high currents and offers long service life. It is preservation free and provides good electrical consistency. It works well at low temperatures and has a low self discharge. It can charge up to five times more rapidly than the flooded type batteries. AGM offers a discharge of 80 percent and on the other hand flooded is specified at 50 percent to achieve the same cycle life.

AGM batteries are established in high end vehicles to run high power accessories. AGM products are trembling resistant. AGM is the appropriate battery for expensive motorcycles. Being sealed, AGM reduces acid spilling in an accident, lowers the weight for the same performance and allows setting up at unusual angles. Because of good performance at cold temperatures, AGM batteries are also used for marine, motor home

and robotic application. AGM batteries are sensitive to overcharging. These batteries can be charged to 2.40V/cell without trouble. AGM batteries should be installed away from the engine compartment due to heat resistive nature.

As there are many advantages of AGM batteries but there are few demerits of it. It has higher built-up cost than flooded; it is greatly sensitive to overcharging. It has low precise energy and it must be stored in charged state. If its AGM shield gets break then its efficiency is decreased and its life cycle is reduced. So during its manufacturing this process should be done very sensitively and effectively.

Advantages of AGM over Flooded Deep Cycle Battery

Characteristics	Deep Cycle AGM battery	Flooded Cycle battery
Self Discharge	1 to 3% per month	5-10% per month
Water Addition	Never	Frequently require to maintain electrolyte level
Hydrogen Gas Emissions	Negligible unless severely overcharged.	Significant volume is generated and must be ventilated to prevent explosion.
Electrolyte Spillage	Non spillable in all orientations – electrolyte is retained in AGM separator.	Spills when tilted, inverted, or cracked.
Electrolyte Stratification	No stratification occurs.	Stratification occurs when operated at low charging voltages or in taller batteries.
Tolerance to freezing Temperature	No damage when frozen	Batterydestroyed when frozen

Table 1 : AGM battery vs Flooded battery

CHAPTER 3
DESIGN AND DEVELOPMENT

DESIGN AND DEVELOPMENT

3.1 Design summary

Scilab based motor simulation model will be established. Arduino board will be installed and all its libraries will be updated for Scilab real time processing. Input of motor or transmitted signal is entirely generated through software and a received signal is entirely processed and demodulated within software algorithms. Then this output from Scilab will be fed to Arduino board so that motor should be given input using motor driver to work accordingly. This can be well understood with the help of figure-3.

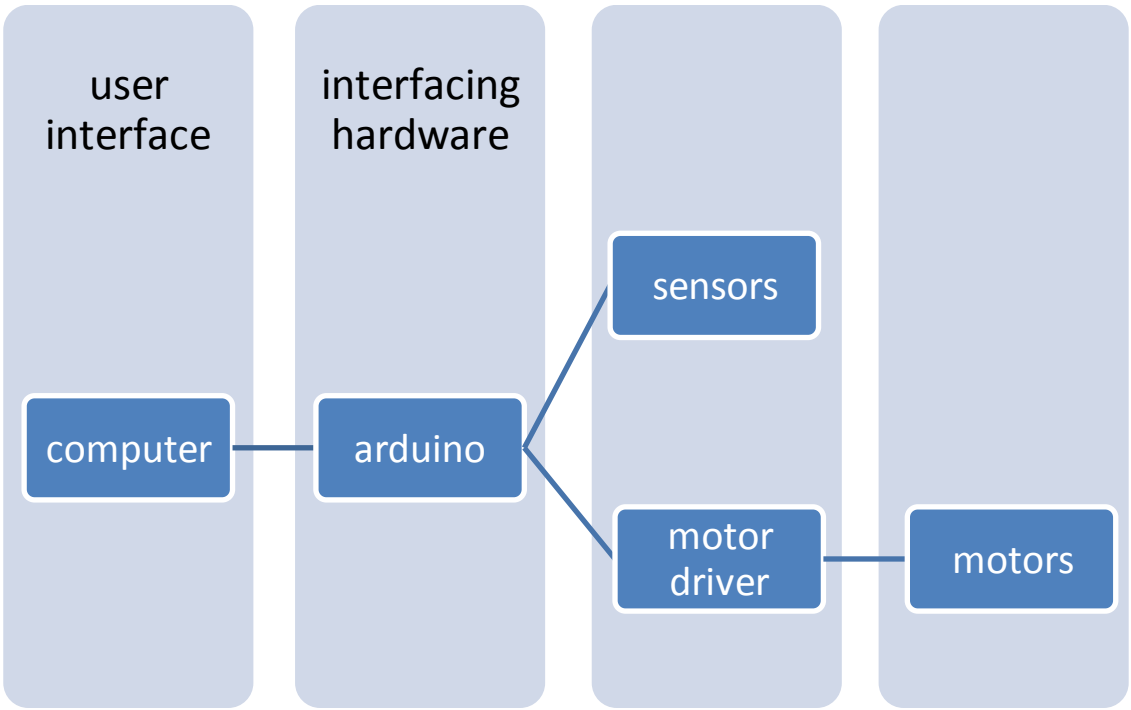

Figure 7 : Design summary

3.2 Mechanical part:

Robotic Arm

- Specifications

Degree of Freedom	
3	
Dimensions	
Base	12″ x 12″ x 6″ (length x Width x Height)
Height	30″
Width	18″
Operating Voltage	
DC-12V	
Range of Base	180^{o}
Lift Weight	
0.5 kg	
Material Required	
Wood	
Unbreakable PVC (Diameter = 3.5″ , 2″)	

Table 1 : Robotic arm specifications

- **Software Design:** The software design of robotic arm is made using Pro-Engineer mechanical software and the results are shown below.

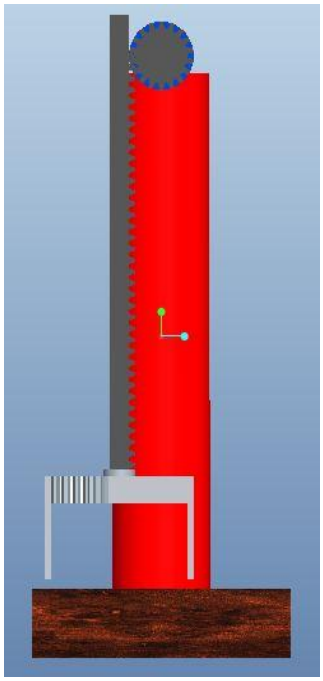

Figure (6a) Front view

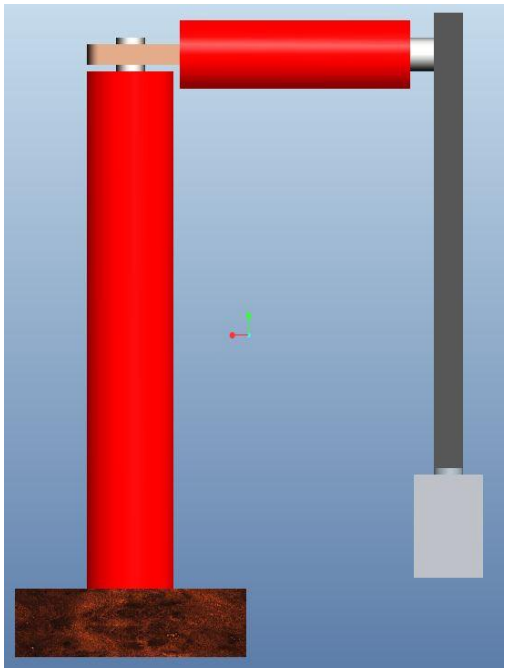

Figure (6b) Right view

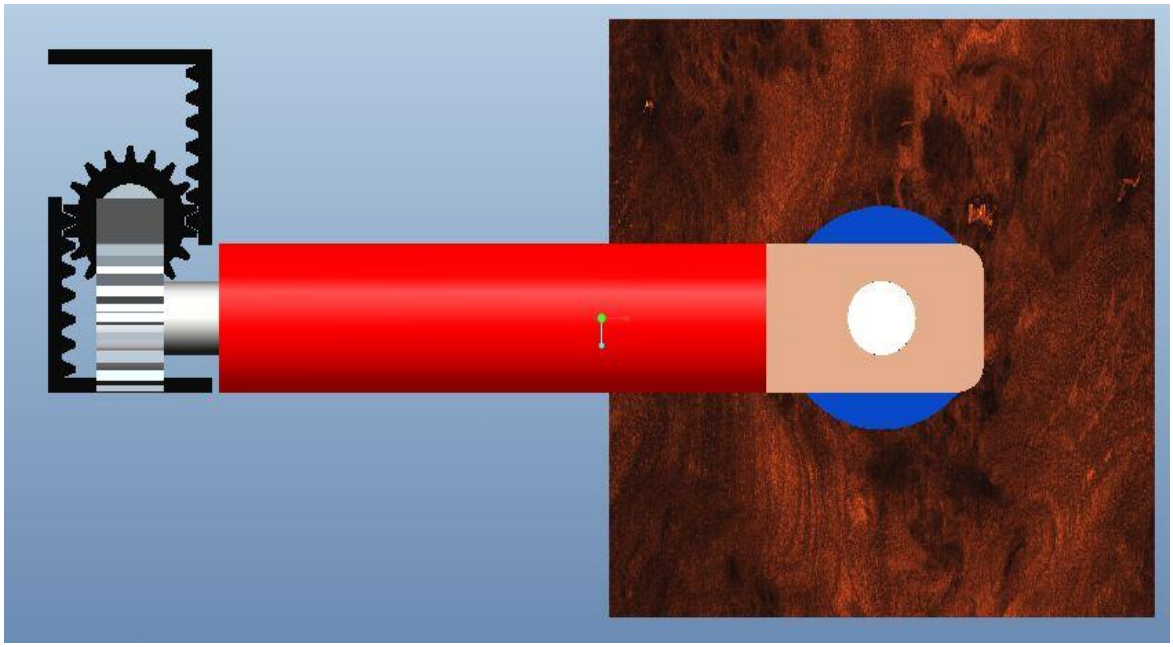

Figure (6c) Top view

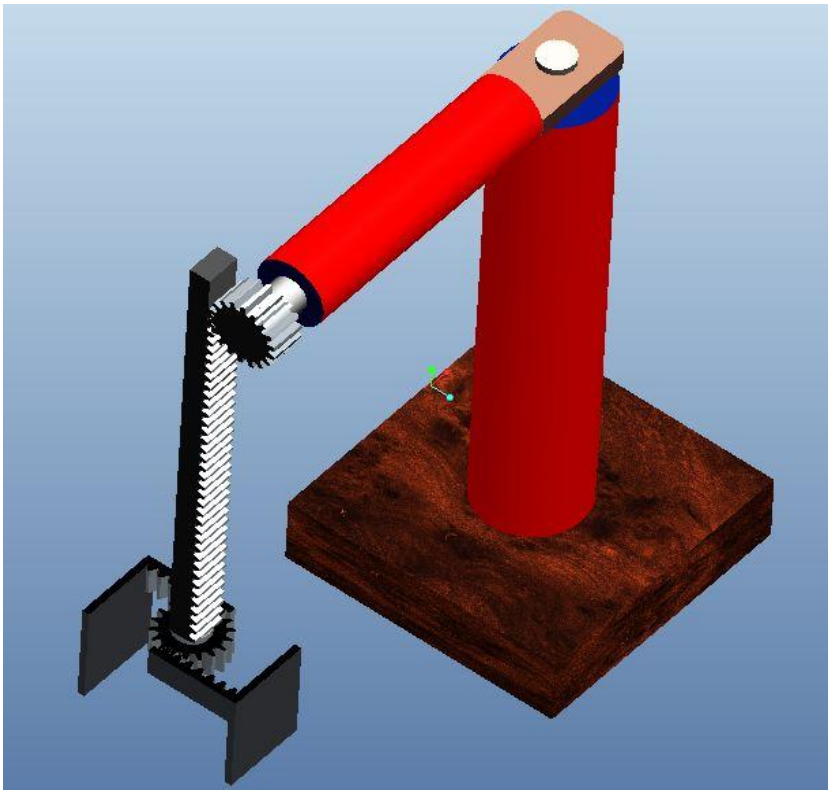

Figure (6d) Standard Orientation

Figure 8 : Robotic arm

Function

Robotic arm will be used to pick the cells of the battery from one conveyor belt and place them into the container on the second conveyor belt.

Working

Robotic arm is controlled by the computer software Scilab. Instructions are given to the motor drivers to drive the motors using arduino board. Robotic arm has limit switches connected at different places which control the position of the robotic arm, when a specific position is attained, it triggers the limit switch, a feedback is sent to the computer and based on that signal and the algorithm implemented in the software, a new set of instructions is given to the robotic arm for further movement.

Conveyor Belt

- **Specifications**

Dimensions	
Length	48 inches
Width	8 inches
Height	8 inches
Max Weight	
1.5 kg	
Speed	
Without Load	90 rpm
With Load	60rpm
Operating Voltage	
DC-12V	
Operating Current	
2 A	
Torque	
3 Nm	
Material Required	
Wood	
Teflon Rollers (1″ diameter)	
Belt	

Table 2 : Conveyor belt specifications

- **Software Design**

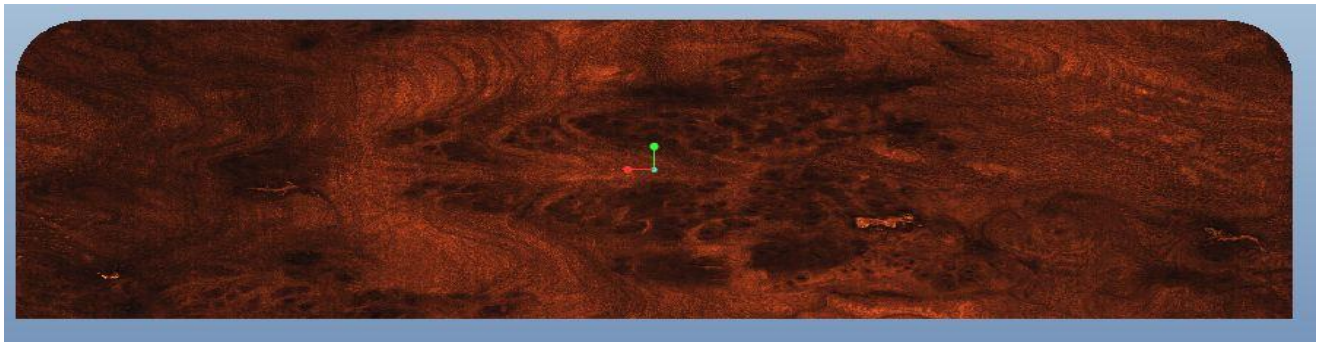

Figure (7a): Front view

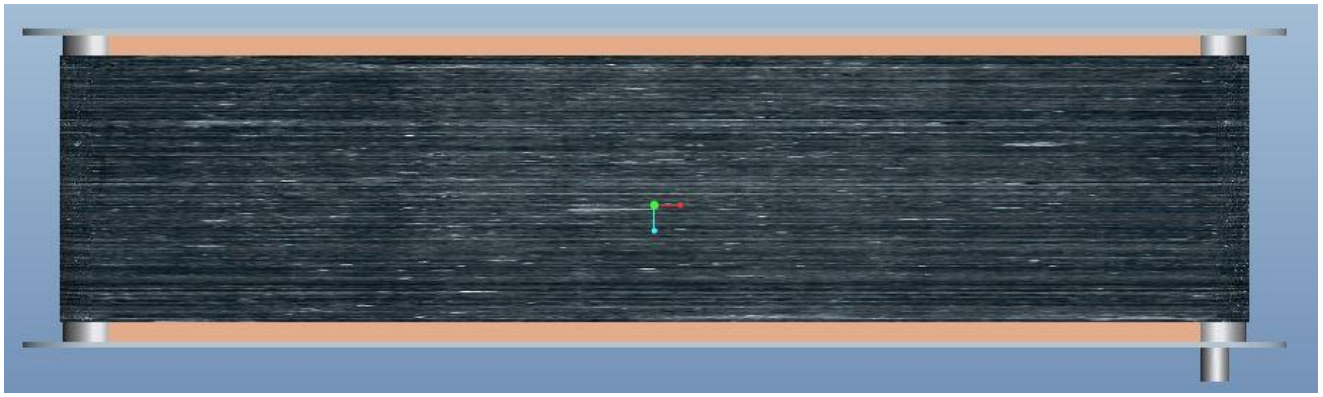

Figure (7b): Top view

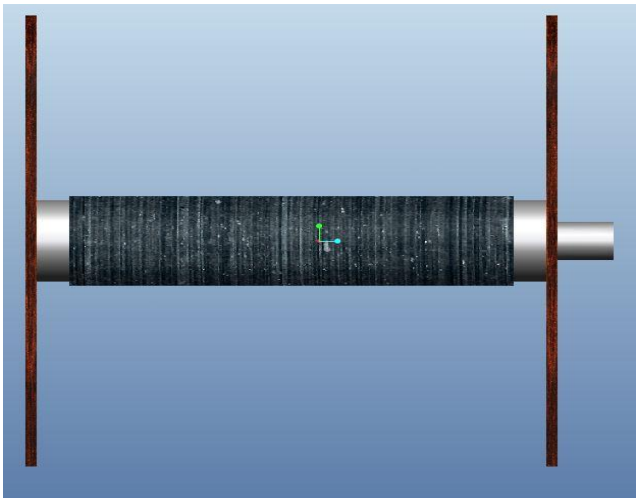

Figure (7c): Side view

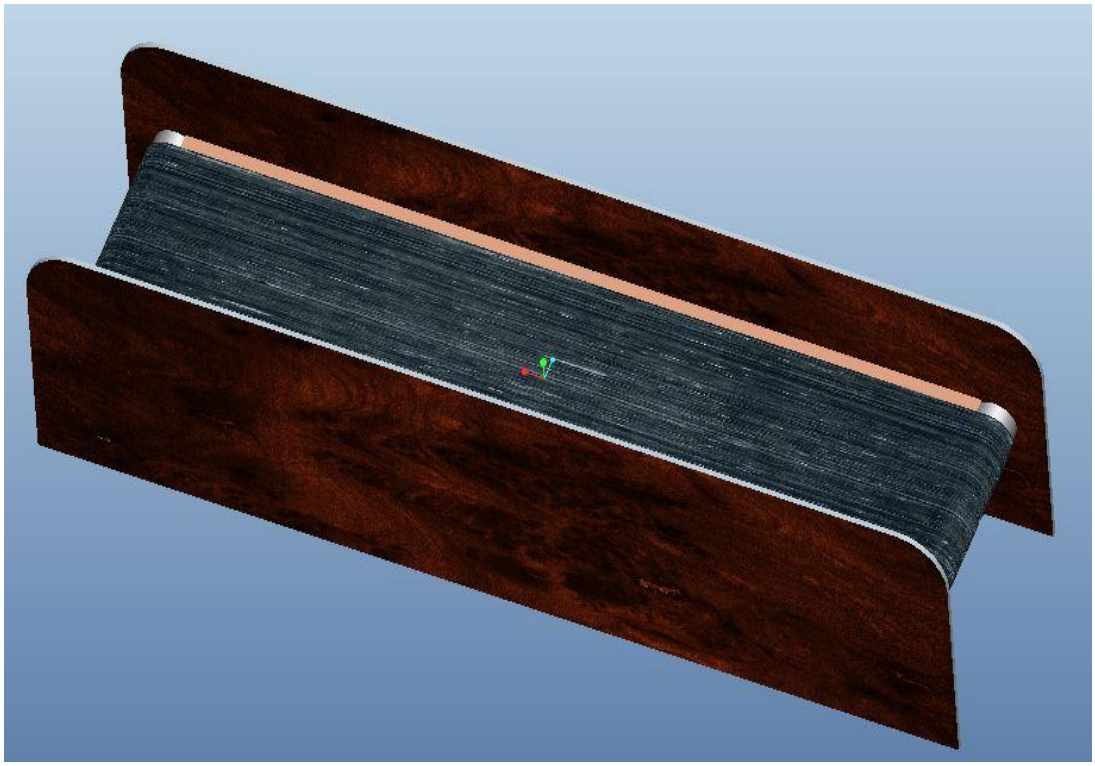

Figure (7d): Standard Orientation

Figure 9 : Conveyor belt

Function

The function of the conveyor belts are to move the cells and the container to a specific location such that the robotic arm can easily pick the cells from one conveyor belt and place them into the container on the other conveyor belt efficiently.

Working

Conveyor belt is also controlled by the computer software Scilab. Infrared switches are connected with the conveyor belt to locate the position of cells and container of the battery. When cells or the container come in front of the IR, a signal is sent to the computer to stop the conveyor instantaneously. Computer in response stops the conveyor. There are two conveyor belts, one would be used for cells and the other would be used for the container.

3.3 Software part

SCILAB

- **Robotic Arm**

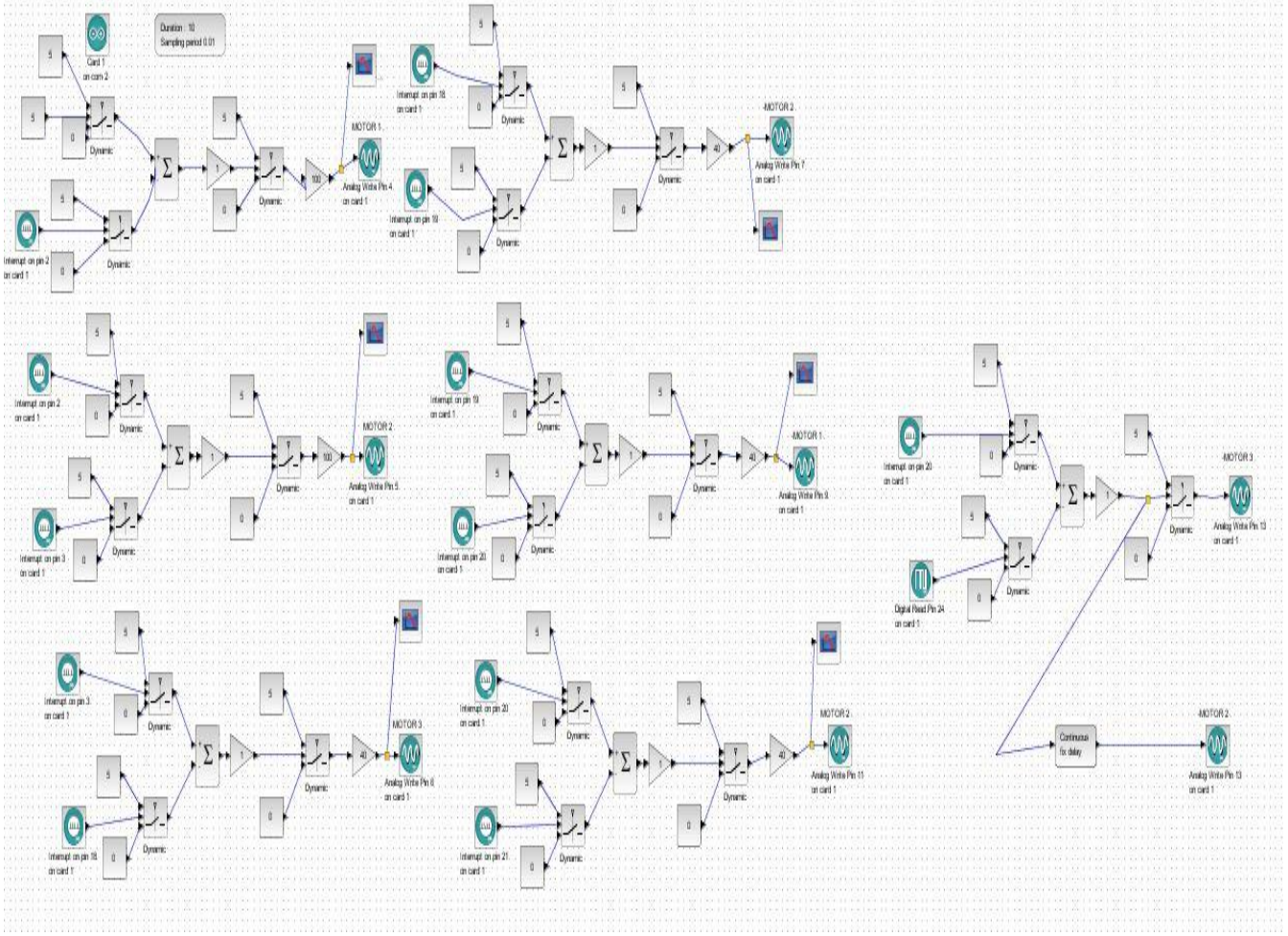

Figure 10 : SCILAB model for robotic arm

➢ **Conveyor Belt**

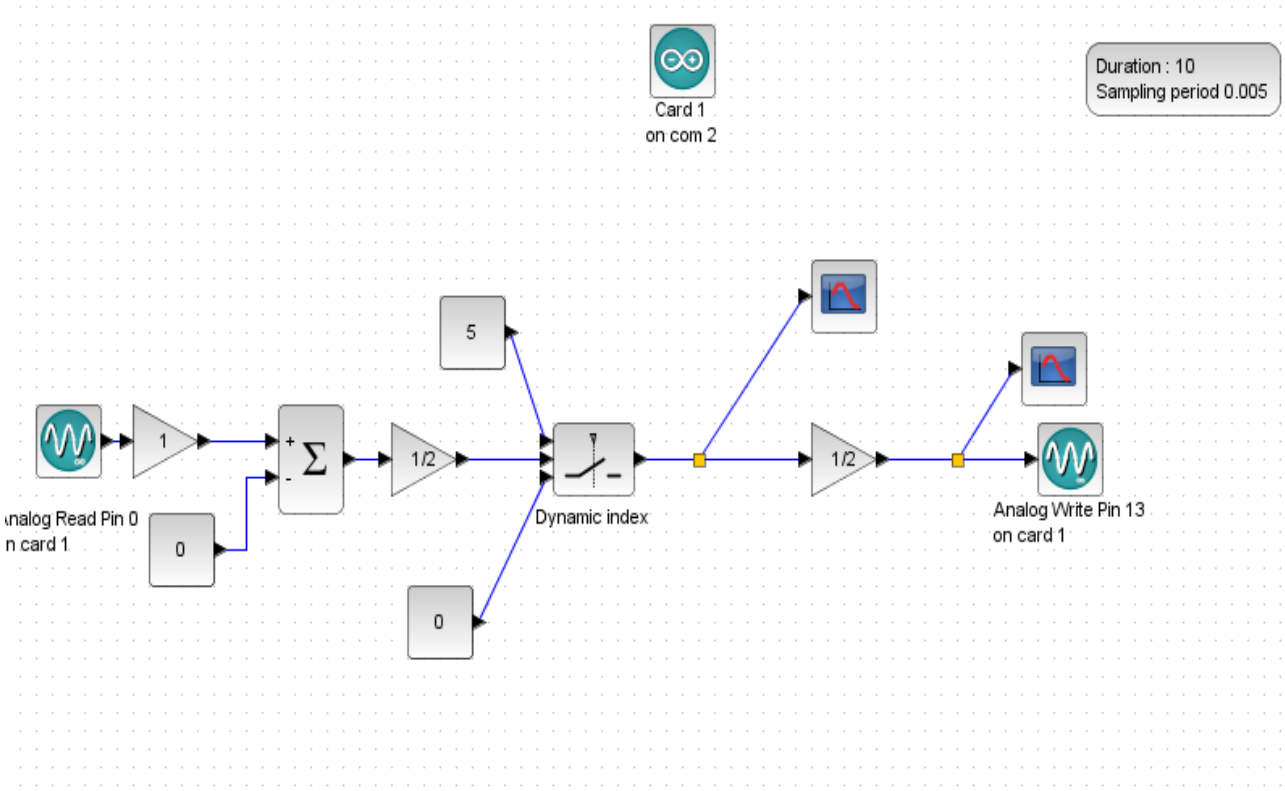

Figure 11 : SCILAB model for conveyor belt

Working

Scilab will receive the feedback signal from robotic arm and conveyor belt. It process the signal and decide when to move the conveyor and when to stop.Scilab also give signal to the motors of the robotic arm to move at appropriate time.

In the scilab model of conveyor belt, analoge read pin 0 will take the input from infrared sensors. Gain blocks are used to calibrate the signals and to remove the noises. Dynamic switch is used as a threshold for the signals, if it is below the threshold output will be 0 and if it is above the threshold output will be 5. This is helpful for computer to determine if the signal is meaningful or not. If the output of the dynamic switch is high it tells the arduino write pin 13 to activate the motor driver of the conveyor belt. If there is nothing infront of infrared receiver the input to the computer is high and conveyor belt will move

continuously, and whenever an object is arrived in front of infrared receiver the input to the computer is low and conveyor belt will be stopped instanteously by the computer.

3.4 Electrical part

- **Arduino Board:**

Figure 12 : Arduino Mega 2560

- **Specification**

Microcontroller	ATmega2560
Operating Voltage	5V recommended 7-12V
Input Voltage limit	DC-20V
Digital I/O Pins	54
Analog Input Pins	16
Flash Memory	256 KB
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Table 3 : Arduino specifications

Function

Arduino is used to interface between Scilab and robotic arm . Arduino toolbox v3_ino is burnt into the microcontroller ,this toolbox is used for real time communication between Scilab and the arduino board. Arduino take input from the switches attached to the robotic arm and send it to the scilab . This signal inform scilab about the current position of the arm .Scilab process the signal and send back signal to the arduino board which gives output to the motor drivers of the robotic arm to move in the desired direction.

- **Motor driver**

Components Required :

Components	Quantity
Relay DC-12V	3
Diode 1N4148	3
Transistor 2N3904	3
Terminal Blocks	6
Infrared Transmitter	1
Infrared Receiver	1

Table 4: Components of motor driver

Robotic Arm motor driver

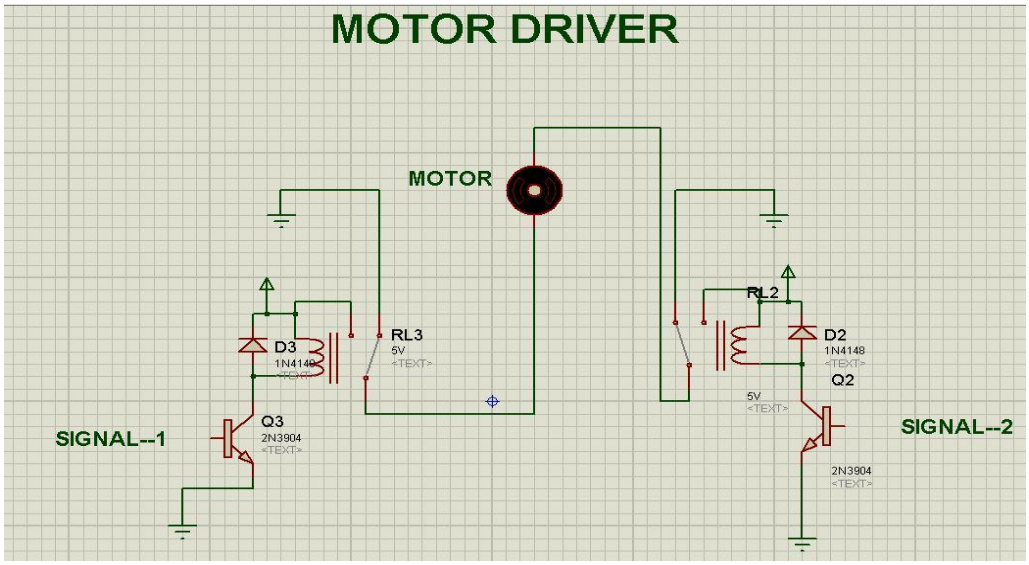

Figure (11a): Robotic Arm Motor Driver

Conveyor Belt motor driver

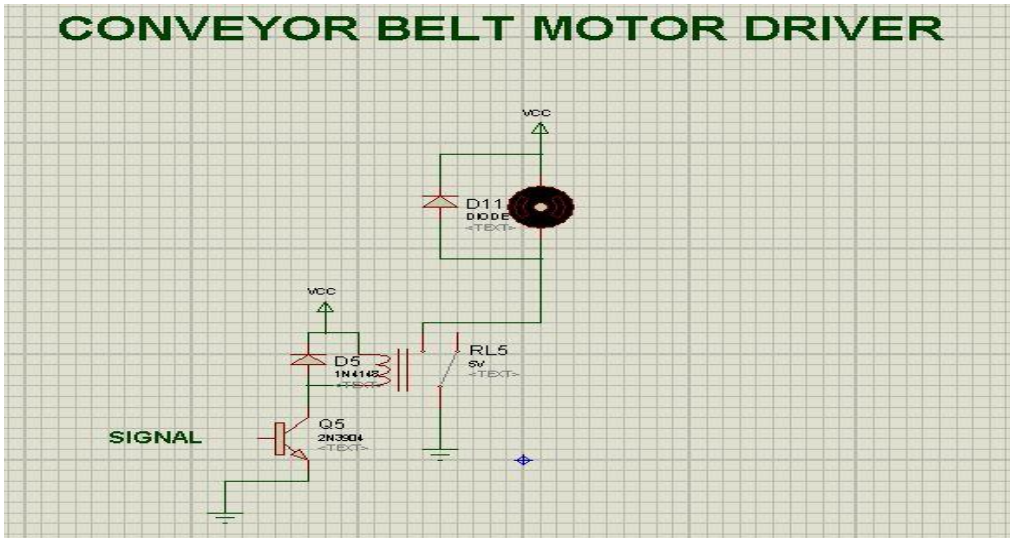

Figure (11b): Conveyor Belt Motor Driver

Figure 13 : Motor Drivers

Function

Max output voltage from the arduino is 5 V with 40 mA current and motor used in conveyor belt and robotic arm requires DC-12V with high current value .The voltage and current supplied by the arduino is not enough to drive the motors. So arduino output is connected to the motor drivers to supply enough voltage and current to drive the motors.

Working

Motor drivers are designed using relays. When there is no signal on any of the transistor output of both relays are connected to the +Vcc and motor does not move (electronic brakes).When there is signal on any of the transistor transistor comes in saturation mode and relay is energized now the output of one of the relay is connected to the +Vcc and the others output is connected to the ground so motor starts rotating.

Infrared Switch

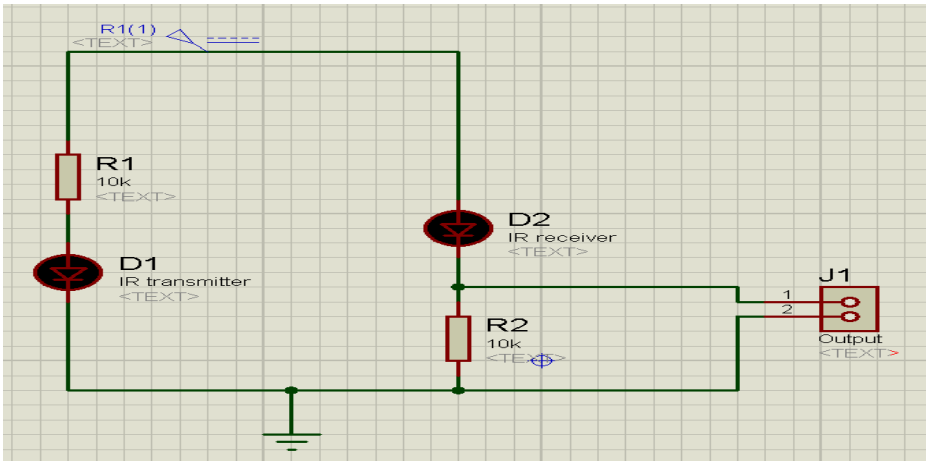

Figure 14 : Infrared Switch

Working

When IR rays fall on the receiver, resistance of the IR sensor is very small so all the voltage is dropped across the resistor R2 shown in the above figure so high signal is send to the Scilab. when cells come in front of the transmitter no rays falls on the receiver and resistance of the IR sensor is very high and no voltage drops across R2 and there is no output at the terminal so low signal is send to the scilab.

- **Motor**

Specifications

Rated Voltage	DC-12V
Rated Torque	3 Nm
No Load Current	2.8
No Load speed	90 rpm
Rated Current	9 A
Rated Speed	60 rpm
Stall Torque	9

Table 5 : Motor specifications

Figure 15: Motor

CHAPTER 4
PROJECT DEVELOPMENT

PROJECT DEVELOPMENT

4.1 Conveyor Belt

Figure 16 : Conveyor Belt

4.2 Robotic Arm

Figure 17 : Robotic Arm

4.3 Testing

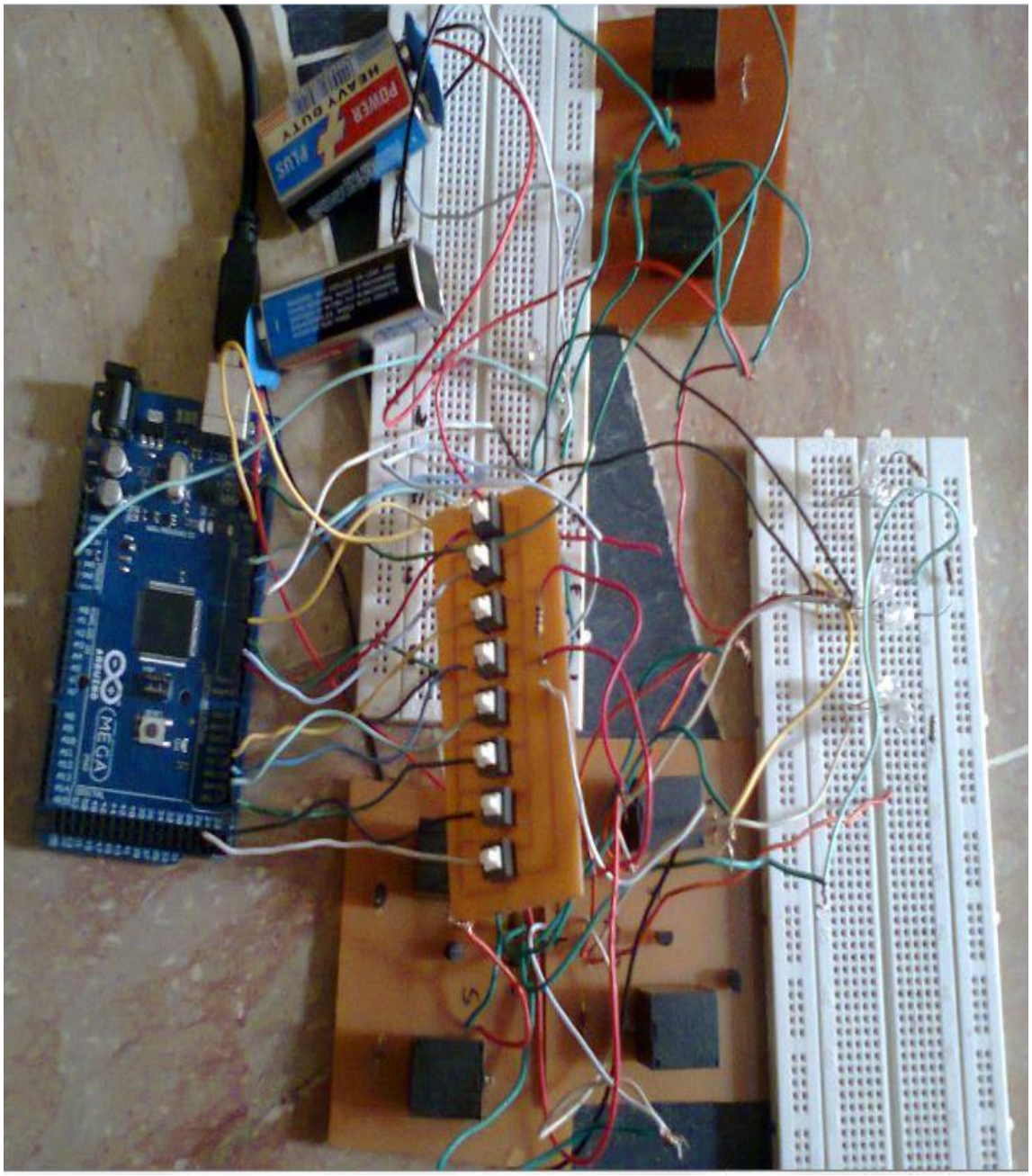

Figure 18 : Scilab testing circuit

4.4 Motor Driver

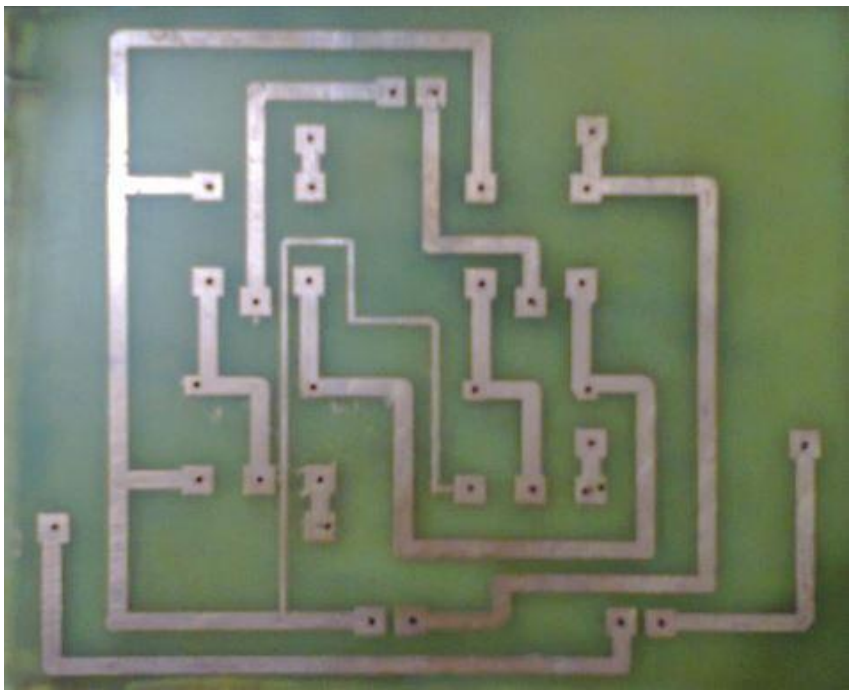

Figure 19 : Motor Driver pcb design

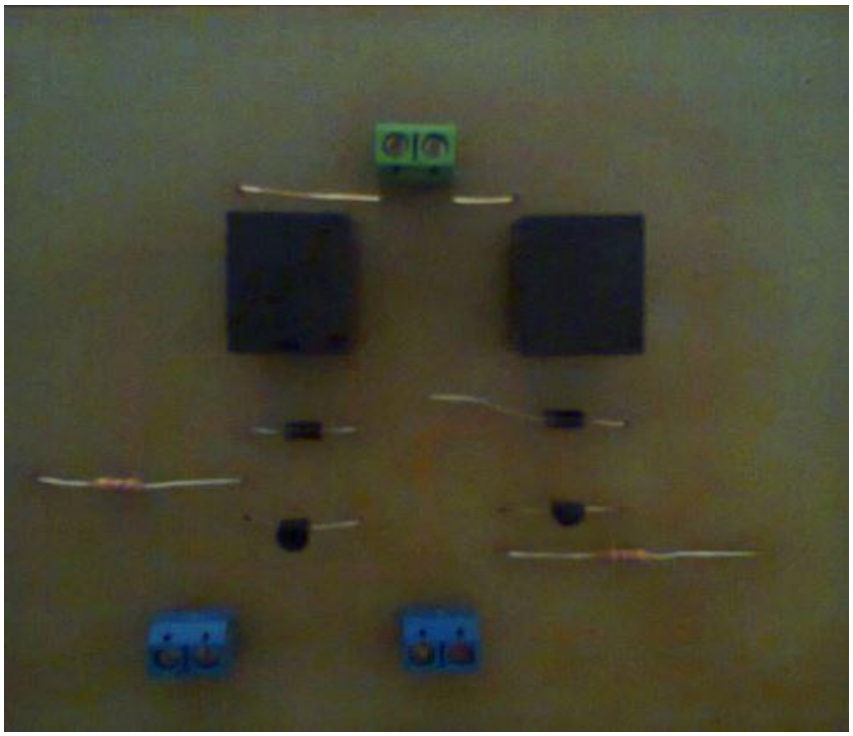

Figure 20 : Robotic Arm Motor Driver

4.5 Infrared Transmitter

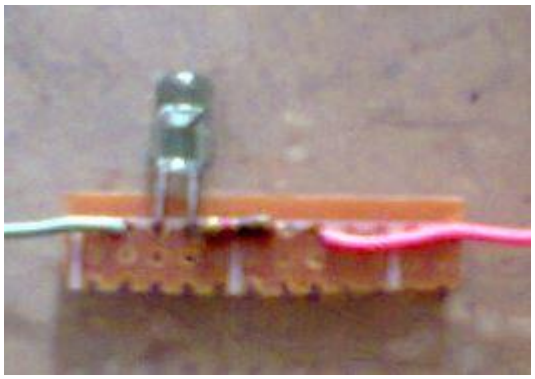

Figure 21 : Infrared Transmitter

Receiver

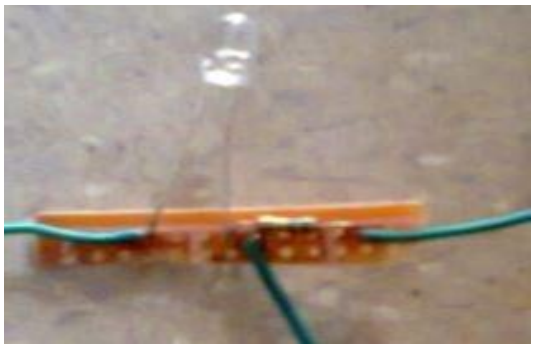

Figure 22 : Infrared Receiver

Conveyor Motor Driver

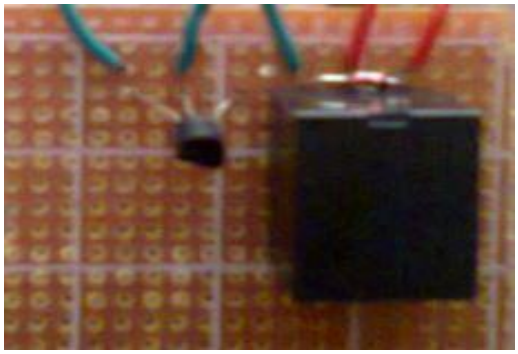

Figure 23 : Conveyor Motor Driver

4.6 Faulty Works

The following figures are of faulty motor drivers. They were not etched properly and there were some design problems which are solved by redesigning the PCB.

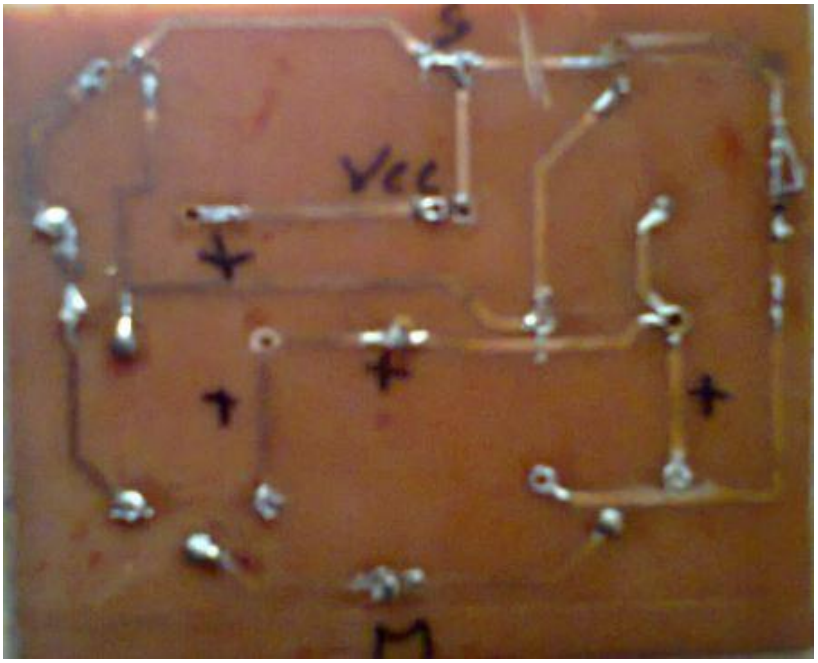

Figure (22 a): Improperly Designed

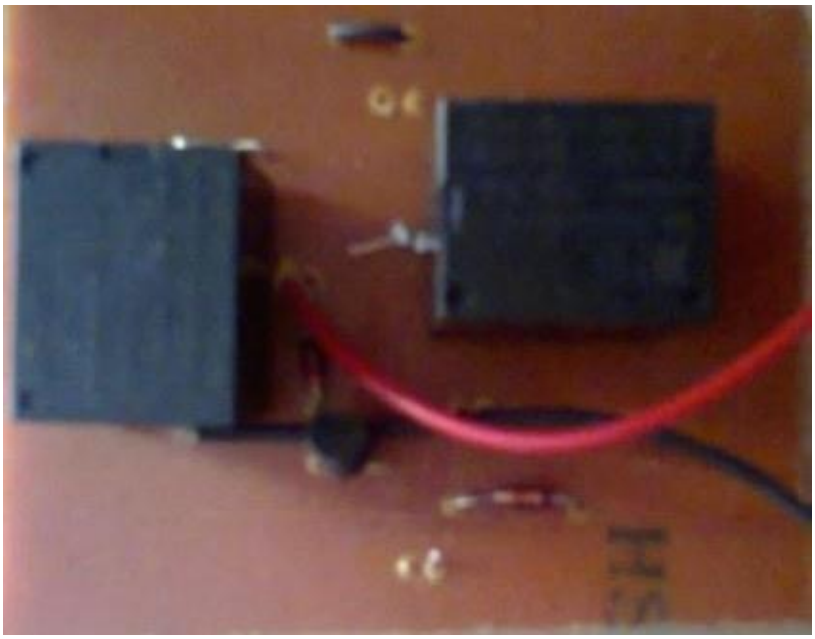

Figure (22 a): Improperly Designed

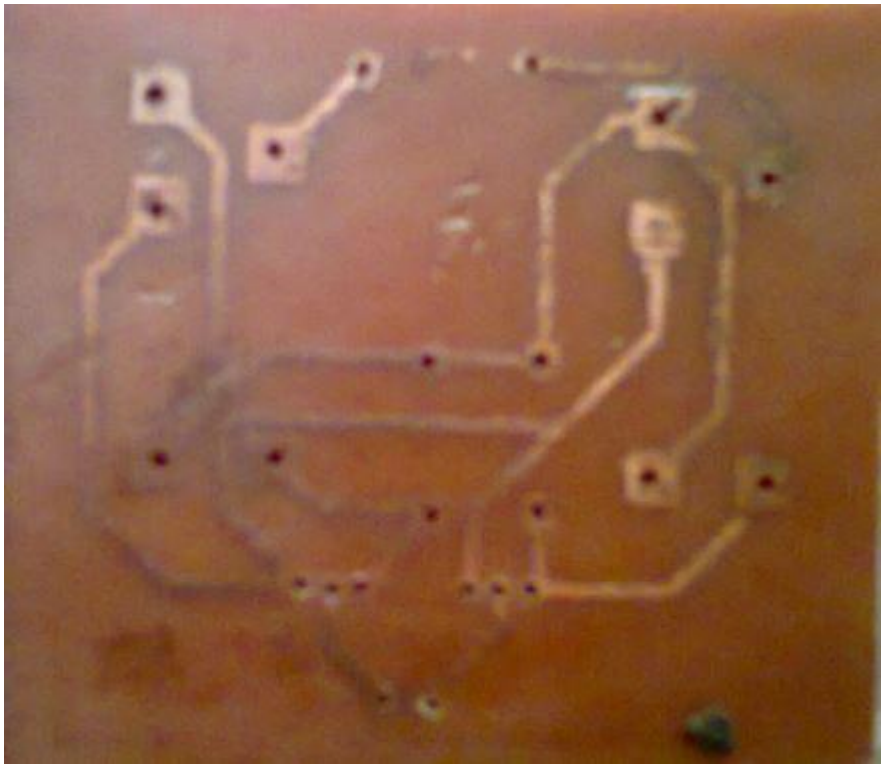

Figure (22d): Improperly etched

Figure 24 : Faulty PCBs

CHAPTER 5
RECOMMENDATIONS FOR FUTURE WORK

RECOMMENDATIONS FOR FUTURE WORK

- Robotic arm is an efficient and most reliable way of automation for manufacturing process in industries.
- Current prototype is designed to place one type of cell in a battery so this can be enhanced to pick all type of cells and efficiently place them into the container provided we have high-quality mechanical structure and good actuators to drive them.
- Robotic arm can be used for packing purposes.
- Links can be fabricated using some other material which is sturdy enough and light in weight.
- We can use image processor so that robotic arm can detect the faulty cells and remove it from the conveyor belt.

CHAPTER 6
CONCLUSION

CONCLUSION

6.1 Overview

Companies today are a goal of cut throat contest. If a company wants to stay in business, it has to supply improved and more well organized products and services in smaller time than their competitors the opposite of this would indicate death for the business. So basically in businesses today, it's all about superiority and time.

The traditional method of inserting cells into the container is no more acceptable because it reduces the efficiency of the battery.

Automation develops a tremendously skilled process designed to keep the manufacturer's quality control values. Besides the consequences of introducing automation results in less no of injuries at the manufacturing line, reduce human handling that improve hygiene thus improving quality promises as every cell is properly inserted in the container .All of these factors taken together leads to major effort and increasing profits.

In summary automating cell insertion process improves efficiency and ensures exact and perfect product developing. It is a straightforward and economical solution to improve the product and companies reputation.

6.2 Objective Achieved

We have successfully designed and manufactured the robotic arm and the conveyor belts

We have successfully interfaced arduino with the computer and configured it with the SCILAB using arduino toolbox v3_ino.

We have established SCILAB model in XCOS which can control the movements of robotic arm and the conveyor belts.

We have designed, simulated and fabricated the motor drivers and infrared sensors.

CHAPTER 7
BIBLIOGRAPHY

BIBLIOGRAPHY

7.1 Previous Work Done

Previously different groups have done great work on Robotic arm for different purposes. Usually they used to work with the help of IC's and microcontroller for only a fixed routine of work. Once they were programmed and then they do only that specified work.

[1] Vhtomitsy, Valer , "Robot Arm", Department of Computer Science and Engineering, Politehnica University of Timi, 2006

[2] MeganMadariaga, Asma Ali, "Assistive Robotic Arm" Hampton Elementary School

[3] MOHAMED NAUFAL BIN OMA, "Pick and Place Robotic Arm Controlled By Computer", UniversityTechnical Malaysia Melaka, April 2007

[4] Falak k. Dalal, "Pick & place Robotic Arm", DEPARTMENT OF MECHATRONICS, U.V. PATEL COLLEGE OF ENGINEERING GANPAT UNIVERSITY, KHERVA, MAY-JUNE -2008

[5] NikhilSawake, "Intelligent Robotic Arm" Visvesvaraya National Institute of Technology, Nagpur July 2013

7.2 Reference

[1] SCILAB (Real time toolbox for SCILAB)

[2] G.W.Younkin,Industrial Servo Control Systems – Fundamentals and Applications –Second Edition ,Taylor and Francis , 2007

[3] H.J.Luinge , P.H. Veltinkb and C.T.M Baten .Ambulatorymeasurment of arm orientation .Journal of Biomechanics .University of Twente,2005.

[4] http://www.coolest-gadgets.com/20091111/darth-vader-robotic-arm

[5] http://en.wikipedia.org/wiki/Robot

[6] D. Ng, "Robot Hand/Arm," [Online]. http://www.androidworld.com/prod61.htm (Accessed: 13 April 2013).

[7] M.W. Spong, S. Hutchison & M. Vidyasagar, Robot dynamics and control. 2nd ed., San Fransisco: Academia.edu, 2004,

[8] arduino.cc/en/uploads/Main/arduino-mega2560-schematic.pdf

APPENDICES

APPENDIX-A
CODE

```
/* this file should be used with the SCILAB arduino

  Conversion ascii -> number
  48->'0' ... 57->'9' 58->':' 59->';' 60->'<' 61->'=' 62->'>' 63->'?' 64->'@'
  65->'A' ... 90->'Z' 91->'[' 92->'\' 93->']' 94->'^' 95->'_' 96->'`'
  97->'a' ... 122->'z'

  Dan0 or Dan1 : attach digital pin n (ascii from 2 to b) to input (0) or output (1)
  Drn : read digital value (0 or 1) on pin n (ascii from 2 to b)
  Dwn0 or Dwn1 : write 1 or 0 on pin n
  An    : reads analog pin n (ascii from 0 to 19)
  Wnm  : write analog value m (ascii from 0 to 255) on pin n (ascii from 0 to 19)
  Sa1 or Sa2 : Attach servo 1 (digital pin 9) or 2 (digital pin 10)
  Sw1n or Sw2n : moves servo 1 or servo 2 to position n (from ascii(0) to ascii(180))
  Sd1 or Sd2 : Detach servo 1 or 2

  Generic DC_Motor
  Cijkl : setup for generic DCmotor number i (1 to 4), PW1 on pin number j, PWM2 or
direction on pin number k, mode=l
        l=0 for L293 (2 PWM) and l=1 for L298 (1PWM + 1 bit for direction)
  Mijk  : sets speed for generic DCmotor number i, j=0/1 for direction, k=ascii(0) ..
ascii(255)
  Mir   : releases motor i (r=release)

  Generic Interrupt counter
  Iai   : activate counter on INT number i (i=ascii(2 or 3 or 18 or 19 or 20 or 21)
  Iri   : release counter on INT number i
  Ipi   : read counter on INT number i
  Izi   : reset counter on INT number i

  Generic Encoder
  Eajkl: activate encoder on channelA on INT number j (j=ascii(2 or 3 or 18 or 19 or 20
or 21) et channelB on pin k or INT number k (k=ascii(0)..ascii(53))
        and l=1 or 2 or 4 for 1x mode (count every rising of chA) or 2x mode (count every
change statement of chA)
        or  4x mode (every change statement of chA et chB)
  Eri   : release encoder on INTi
  Epi   : read position of encoder on INTi
```

```
  Ezi   : reset value of encoder on INTi position

  R0    : sets analog reference to DEFAULT
  R1    : sets analog reference to INTERNAL
  R2    : sets analog reference to EXTERNAL

  */

#include <Servo.h>

/* define internal for the MEGA as 1.1V (as as for the 328)  */
#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
#define INTERNAL INTERNAL1V1
#endif

/* create and initialize servos                        */
Servo servo1;
Servo servo2;

/* Generic motors */
int dcm1_pin1,dcm1_pin2,dcm1_mode;
int dcm2_pin1,dcm2_pin2,dcm2_mode;
int dcm3_pin1,dcm3_pin2,dcm3_mode;
int dcm4_pin1,dcm4_pin2,dcm4_mode;

// Generic encoder
/* Encoders initialisation */
// volatile declare as those variables will change in interrupts
volatile long int encoder_0_position = 0,encoder_1_position = 0, encoder_2_position = 0,
encoder_3_position = 0, encoder_4_position = 0, encoder_5_position = 0;
int encoder_0_int2 ;        // Pin used for encoder0 chanel B : define from scilab
int encoder_1_int2 ;        // Pin used for encoder1 chanel B : define from scilab
int encoder_2_int2 ;        // Pin used for encoder2 chanel B : define from scilab
int encoder_3_int2 ;        // Pin used for encoder3 chanel B : define from scilab
int encoder_4_int2 ;        // Pin used for encoder4 chanel B : define from scilab
int encoder_5_int2 ;        // Pin used for encoder5 chanel B : define from scilab
int encoder_num, encoder_int2;
int corresp[6]={2,3,21,20,19,18}; //Correspondance beetween interrupt number and pin
number
```

```
//Generic counter
volatile long int
counter_0=0,counter_1=0,counter_2=0,counter_3=0,counter_4=0,counter_5=0;

int initiat=1;

void setup() {
  /* initialize serial                              */
  Serial.begin(115200);

}

void loop() {

  /* variables declaration and initialization              */

  static int  s   = -1;    /* state                        */
  static int  pin = 13;    /* generic pin number           */
  static int  dcm =  4;    /* generic dc motor number      */

  int  val =  0;          /* generic value read from serial */
  int  agv =  0;          /* generic analog value         */
  int  dgv =  0;          /* generic digital value        */
  static int  enc   = 1;    /* encoder number 1 (or 2 for Arduino mega)    */

  while (Serial.available()==0) {}; // Waiting char
  val = Serial.read();

//  if (val==0){// version
//    Serial.print('v3');
//    val=-1;
//  }
  //case A -> Analog
  if (val==65){//A -> Analog read
    while (Serial.available()==0) {}; // Waiting char
//    val=Serial.read();
//    if (val==114){ //'r'-> read pin
//      while (Serial.available()==0) {}; // Waiting char
      val=Serial.read();
      if (val>47 && val<67) { //from pin 0, to pin 19
```

```
        pin=val-48; //number of the pin
        agv=analogRead(pin);
        //Serial.println(agv);
        Serial.write((uint8_t*)&agv,2); /* send binary value via serial  */
      }
      val=-1;
}
else if (val==87){//W -> Analog write
    while (Serial.available()==0) {}; // Waiting char
    val=Serial.read();
       if (val>47 && val<67) { //from pin 0 to pin 19
          pin=val-48; //number of the pin
          while (Serial.available()==0) {}; // Waiting char
          val=Serial.read();
          analogWrite(pin,val);
       }
        val=-1;
    }
//}

//case D -> Digital
else if (val==68){//D -> Digital pins
  while (Serial.available()==0) {}; // Waiting char
  val=Serial.read();
  if (val==97){ //'a'-> declare pin
     while (Serial.available()==0) {}; // Waiting char
     val=Serial.read();
     if (val>49 && val<102) {
       pin=val-48;
       while (Serial.available()==0) {}; // Waiting char
       val=Serial.read();
       if (val==48 || val==49) {
          if (val==48){//'0' -> input
             pinMode(pin,INPUT);
          }
          else if (val==49){//'1' -> output
             pinMode(pin,OUTPUT);
          }
       }
     }
```

```
    }
    if (val==114){ //'r'-> read pin
      while (Serial.available()==0) {}; // Waiting char
      val=Serial.read();
      if (val>49 && val<102) {
        pin=val-48; //number of the digital pin
        dgv=digitalRead(pin);
//        Serial.println(dgv);
        Serial.print(dgv);
      }
    }
    if (val==119){ //'w'-> write pin
      while (Serial.available()==0) {}; // Waiting char
      val=Serial.read();
      if (val>49 && val<102) {
        pin=val-48; //number of the digital pin
        while (Serial.available()==0) {}; // Waiting char
        val=Serial.read();
        if (val==48 || val==49) { // 0 or 1
          dgv=val-48;
          digitalWrite(pin,dgv);
//          Serial.println(dgv);
        }
      }
    }
   val=-1;

  }
  //case S -> servomotor
  else if (val==83){
    while (Serial.available()==0) {}; // Waiting char
    val=Serial.read();
    if (val==97){ //'a'-> declare servo
      while (Serial.available()==0) {}; // Waiting char
      val=Serial.read();
      if (val==49 || val==50) { //servo 1 or 2
        pin=val-48; //number of the servo
        if (pin==1) {
          servo1.attach(9);
          servo1.write(0);
```

```
//            agv=servo1.read();
//            Serial.println(agv);
          }
          if (pin==2) {
            servo2.attach(10);
            servo2.write(0);
//            agv=servo2.read();
//            Serial.println(agv);
          }

        }
      }
      if (val==100){ //'d'-> detach servo
        while (Serial.available()==0) {}; // Waiting char
        val=Serial.read();
        if (val==49 || val==50) { //servo 1 or 2
          pin=val-48; //number of the servo
          if (pin==1) {servo1.detach(); }
          if (pin==2) {servo2.detach(); }
        }
      }
      if (val==119){ //'w'-> write pin
        while (Serial.available()==0) {}; // Waiting char
        val=Serial.read();
        if (val==49 || val==50) { //servo 1 or 2
          pin=val-48; //number of the servo
          while (Serial.available()==0) {}; // Waiting char
          val=Serial.read();
          if (val>=0 && val<=180){
            if (pin==1) {
              servo1.write(val);
//              agv=servo1.read();
//              Serial.println(agv);
            }
            if (pin==2) {
              servo2.write(val);
//              agv=servo2.read();
//              Serial.println(agv);
            }
          }
```

```
      }
    }
   val=-1;

  }

  //case I -> Interrupt
  else if (val==73){
     /* ASKING ACTIVATION OF AN COUNTER     */
     while (Serial.available()==0) {}; // Waiting char
     val=Serial.read();
     if (val==97) { //a = activation
        while (Serial.available()==0) {};              // Waiting char
        val=Serial.read();                          // Read int_number (must be 0 or 1 on UNO /
1 to 5 on MEGA) : int_number set to encoder number
        pinMode(corresp[val],INPUT);              // set interrupt pin as input
        if      (val == 0) {attachInterrupt(val, counter_0_change, RISING);counter_0=0;}
//counter INT0
        else if  (val == 1) {attachInterrupt(val, counter_1_change, RISING);counter_1=0;}
//counter INT1
        else if  (val == 2) {attachInterrupt(val, counter_2_change, RISING);counter_2=0;}
//counter INT2
        else if  (val == 3) {attachInterrupt(val, counter_3_change, RISING);counter_3=0;}
//counter INT3
        else if  (val == 4) {attachInterrupt(val, counter_4_change, RISING);counter_4=0;}
//counter INT4
        else if  (val == 5) {attachInterrupt(val, counter_5_change, RISING);counter_5=0;}
//counter INT5
     }
     /* ASKING POSITION OF A COUNTER     */
     if (val==112) { //p = sending counting value
        while (Serial.available()==0) {};   // Waiting char
        val = Serial.read()   ;            //reading next value = counter number
        if      (val==0){ Serial.write((uint8_t*)&counter_0,4); }// asking counter 0
        else if (val==1){ Serial.write((uint8_t*)&counter_1,4); }// asking counter 1
        else if (val==2){ Serial.write((uint8_t*)&counter_2,4); }// asking counter 2
        else if (val==3){ Serial.write((uint8_t*)&counter_3,4); }// asking counter 3
        else if (val==4){ Serial.write((uint8_t*)&counter_4,4); }// asking counter 4
        else if (val==5){ Serial.write((uint8_t*)&counter_5,4); }// asking counter 5
     }
```

```
    /* ASKING RELEASE OF AN INTERRUPT   */
    if (val==114) { //r = release counter
      while (Serial.available()==0) {};       // Waiting char
      val = Serial.read();                    //reading next value = counter number
      detachInterrupt(val);                    // Detach interrupt on chanel A of counter
num=val
      if (val==0)       { counter_0=0;}       // Reset counter
      else if (val==1)  { counter_1=0;}       // Reset counter
      else if (val==2)  { counter_2=0;}       // Reset counter
      else if (val==3)  { counter_3=0;}       // Reset counter
      else if (val==4)  { counter_4=0;}       // Reset counter
      else if (val==5)  { counter_5=0;}       // Reset counter
    }
    /* ASKING RESET VALUE OF AN COUNTER     */
    if (val==122) { //z set to zero
      while (Serial.available()==0) {};       // Waiting char
      val = Serial.read();                    //reading next value = counter number
      if (val==0)       { counter_0=0;}       // Reset counter
      else if (val==1)  { counter_1=0;}       // Reset counter
      else if (val==2)  { counter_2=0;}       // Reset counter
      else if (val==3)  { counter_3=0;}       // Reset counter
      else if (val==4)  { counter_4=0;}       // Reset counter
      else if (val==5)  { counter_5=0;}       // Reset counter
     }
   val=-1;

 }

 //case E -> Encoder
 else if (val==69){
    /*Generic encoder functions */
    while (Serial.available()==0) {};
    val=Serial.read();
    /* ASKING ACTIVATION OF AN ENCODER     */
    if (val==97) {                            //activation
      while (Serial.available()==0) {};             // Waiting char
      encoder_num=Serial.read();                    // Read int_number (must be 0 or 1 on
UNO / 1 to 5 on MEGA) : int_number set to encoer number
      pinMode(corresp[encoder_num],INPUT);          // set interrupt pin as input
      while (Serial.available()==0) {};             // Waiting char
```

```
      encoder_int2=Serial.read();                  // Read int2 (must be a digital PIN with interrupt or not : depends on mode)
                                                   // no declaration for the moment : wait for encoder mode
      while (Serial.available()==0) {};            // Waiting char
      int mode = Serial.read()-48;                 // Read mode 1 ou 2 (1 counting only rising of chA, 2 counting rising and falling)
      if (mode == 4) {                             // mode 4x : 2 cases : chA=pin2 / chB=pin3 or chA=pin3/chB=pin2 [Uno retriction]
        pinMode(corresp[encoder_int2],INPUT);      // set interrupt number as input
      } else {
        pinMode(encoder_int2,INPUT);               // set pin as input
      }

      if (encoder_num == 0) {                      //encoder INT0
       encoder_0_position=0;                       // Reset position
       if (mode==4) {
        encoder_0_int2=corresp[encoder_int2];      // Save pin of second interruption
        attachInterrupt(encoder_num , encoder_change_m4_A0, CHANGE); // Attach interrupt on chanel A change
        attachInterrupt(encoder_int2, encoder_change_m4_B0, CHANGE); // Attach interrupt on chanel B change
       } else if (mode==2) {
        encoder_0_int2=encoder_int2;
        attachInterrupt(encoder_num, encoder_0_change_m2, CHANGE); // Attach interrupt on chanel A change
       } else if (mode==1) {
        encoder_0_int2=encoder_int2;
        attachInterrupt(encoder_num, encoder_0_change_m1, RISING); // Attach interrupt on chanel A rising
       }
      } else if (encoder_num == 1) {               //encoder INT1
       encoder_1_position=0;                       // Reset position
       if (mode==4) {
        encoder_1_int2=corresp[encoder_int2];      // Save pin of second interruption
        attachInterrupt(encoder_num , encoder_change_m4_A1, CHANGE); // Attach interrupt on chanel A change
        attachInterrupt(encoder_int2, encoder_change_m4_B1, CHANGE); // Attach interrupt on chanel B change
       } else if (mode==2) {
```

```
          encoder_1_int2=encoder_int2;
          attachInterrupt(encoder_num, encoder_1_change_m2, CHANGE); // Attach
interrupt on chanel A change
        } else if (mode==1) {
          encoder_1_int2=encoder_int2;
          attachInterrupt(encoder_num, encoder_1_change_m1, RISING); // Attach
interrupt on chanel A rising
        }
      } else if (encoder_num == 2) {                  //encoder INT2
        encoder_2_position=0;                         // Reset position
        if (mode==4) {
          encoder_2_int2=corresp[encoder_int2];          // Save pin of second interruption
          attachInterrupt(encoder_num , encoder_change_m4_A2, CHANGE); // Attach
interrupt on chanel A change
          attachInterrupt(encoder_int2, encoder_change_m4_B2, CHANGE); // Attach
interrupt on chanel B change
        } else if (mode==2) {
          encoder_2_int2=encoder_int2;
          attachInterrupt(encoder_num, encoder_2_change_m2, CHANGE); // Attach
interrupt on chanel A change
        } else if (mode==1) {
          encoder_2_int2=encoder_int2;
          attachInterrupt(encoder_num, encoder_2_change_m1, RISING); // Attach
interrupt on chanel A rising
        }
      } else if (encoder_num == 3) {                  //encoder INT3
        encoder_3_position=0;                         // Reset position
        if (mode==4) {
          encoder_3_int2=corresp[encoder_int2];          // Save pin of second interruption
          attachInterrupt(encoder_num , encoder_change_m4_A3, CHANGE); // Attach
interrupt on chanel A change
          attachInterrupt(encoder_int2, encoder_change_m4_B3, CHANGE); // Attach
interrupt on chanel B change
        } else if (mode==2) {
          encoder_3_int2=encoder_int2;
          attachInterrupt(encoder_num, encoder_3_change_m2, CHANGE); // Attach
interrupt on chanel A change
        } else if (mode==1) {
          encoder_3_int2=encoder_int2;
```

```
          attachInterrupt(encoder_num, encoder_3_change_m1, RISING); // Attach
interrupt on chanel A rising
         }
       } else if (encoder_num == 4) {                    //encoder INT4
         encoder_4_position=0;                          // Reset position
         if (mode==4) {
           encoder_4_int2=corresp[encoder_int2];          // Save pin of second interruption
           attachInterrupt(encoder_num , encoder_change_m4_A4, CHANGE); // Attach
interrupt on chanel A change
           attachInterrupt(encoder_int2, encoder_change_m4_B4, CHANGE); // Attach
interrupt on chanel B change
         } else if (mode==2) {
           encoder_4_int2=encoder_int2;
           attachInterrupt(encoder_num, encoder_4_change_m2, CHANGE); // Attach
interrupt on chanel A change
         } else if (mode==1) {
           encoder_4_int2=encoder_int2;
           attachInterrupt(encoder_num, encoder_4_change_m1, RISING); // Attach
interrupt on chanel A rising
         }
       } else if (encoder_num == 5) {                    //encoder INT5
         encoder_5_position=0;                          // Reset position
         if (mode==4) {
           encoder_5_int2=corresp[encoder_int2];          // Save pin of second interruption
           attachInterrupt(encoder_num , encoder_change_m4_A5, CHANGE); // Attach
interrupt on chanel A change
           attachInterrupt(encoder_int2, encoder_change_m4_B5, CHANGE); // Attach
interrupt on chanel B change
         } else if (mode==2) {
           encoder_5_int2=encoder_int2;
           attachInterrupt(encoder_num, encoder_5_change_m2, CHANGE); // Attach
interrupt on chanel A change
         } else if (mode==1) {
           encoder_5_int2=encoder_int2;
           attachInterrupt(encoder_num, encoder_5_change_m1, RISING); // Attach
interrupt on chanel A rising
         }
       }
     }
     /* ASKING POSITION OF AN ENCODER     */
```

```
    if (val==112) { //p = sending encoder position
      while (Serial.available()==0) {};   // Waiting char
      val = Serial.read()   ;            //reading next value = encoder number
      if      (val==0){ Serial.write((uint8_t*)&encoder_0_position,4); }// asking encoder 0
position
      else if (val==1){ Serial.write((uint8_t*)&encoder_1_position,4); }// asking encoder
1 position
      else if (val==2){ Serial.write((uint8_t*)&encoder_2_position,4); }// asking encoder
2 position
      else if (val==3){ Serial.write((uint8_t*)&encoder_3_position,4); }// asking encoder
3 position
      else if (val==4){ Serial.write((uint8_t*)&encoder_4_position,4); }// asking encoder
4 position
      else if (val==5){ Serial.write((uint8_t*)&encoder_5_position,4); }// asking encoder
5 position
    }
    /* ASKING RELEASE OF AN ENCODER     */
    if (val==114) { //r = release encoder
      while (Serial.available()==0) {};              // Waiting char
      val = Serial.read();                          //reading next value = encoder number
      detachInterrupt(val);                          // Detach interrupt on chanel A of encoder
num=val
      if (val==0)       { encoder_0_position=0;encoder_0_int2=-1;}       // Reset position
      else if (val==1)  { encoder_1_position=0;encoder_1_int2=-1;}       // Reset position
      else if (val==2)  { encoder_2_position=0;encoder_2_int2=-1;}       // Reset position
      else if (val==3)  { encoder_3_position=0;encoder_3_int2=-1;}       // Reset position
      else if (val==4)  { encoder_4_position=0;encoder_4_int2=-1;}       // Reset position
      else if (val==5)  { encoder_5_position=0;encoder_5_int2=-1;}       // Reset position
      while (Serial.available()==0) {};              // Waiting char
      val = Serial.read();                          // reading next value = encoder number
      detachInterrupt(val);                          // Detach interrupt on chanel B of encoder
num=val (may be the same if mode=1 or 2)
    }
    /* ASKING RESET POSITION OF AN ENCODER     */
    if (val==122) {  // z = encoder position to zero
      while (Serial.available()==0) {};              // Waiting char
      val = Serial.read();                          //reading next value = encoder number
      if (val==0)       { encoder_0_position=0;}       // Reset position
      else if (val==1)  { encoder_1_position=0;}       // Reset position
      else if (val==2)  { encoder_2_position=0;}       // Reset position
```

```
        else if (val==3)  { encoder_3_position=0;}        // Reset position
        else if (val==4)  { encoder_4_position=0;}        // Reset position
        else if (val==5)  { encoder_5_position=0;}        // Reset position
      }
     val=-1;

  }

  //case C -> DCmotor init
  else if(val==67){
    while (Serial.available()==0) {};
    val = Serial.read();
    /* 2nd char = motor number */
    if (val>48 && val<53) {
       dcm=val-48;
       while (Serial.available()==0) {};
       val = Serial.read();
       /* the third received value indicates the pin1 number from ascii(2)=50 to
ascii(e)=101  */
       if (val>49 && val<102) {
         if (dcm==1) dcm1_pin1=val-48;/* calculate motor pin1 */
         if (dcm==2) dcm2_pin1=val-48;/* calculate motor pin1 */
         if (dcm==3) dcm3_pin1=val-48;/* calculate motor pin1 */
         if (dcm==4) dcm4_pin1=val-48;/* calculate motor pin1 */
         pinMode(val-48, OUTPUT); //set pin as output
         analogWrite(val-48,0);   /* DUTY CYCLE */
         while (Serial.available()==0) {};
         val = Serial.read();
         /* the fourth received value indicates the pin2 number from ascii(2)=50 to
ascii(e)=101  */
         if (val>49 && val<102) {
           if (dcm==1) dcm1_pin2=val-48;/* calculate motor pin2 */
           if (dcm==2) dcm2_pin2=val-48;/* calculate motor pin2 */
           if (dcm==3) dcm3_pin2=val-48;/* calculate motor pin2 */
           if (dcm==4) dcm4_pin2=val-48;/* calculate motor pin2 */
           pinMode(val-48, OUTPUT); //set pin as output
           while (Serial.available()==0) {};
           val = Serial.read();
           /* the fifth received value indicates the pin2 number from ascii(2)=50 to
ascii(e)=101 */
```

```
        if (val>47 && val<50) {
          int mode = val-48;
          if (dcm==1) dcm1_mode=mode;/* calculate motor mode */
          if (dcm==2) dcm2_mode=mode;/* calculate motor mode */
          if (dcm==3) dcm3_mode=mode;/* calculate motor mode */
          if (dcm==4) dcm4_mode=mode;/* calculate motor mode */
          //initialization of port
          if(mode==0){//L293
            if (dcm==1) analogWrite(dcm1_pin2,0);   /* DUTY CYCLE */
            if (dcm==2) analogWrite(dcm2_pin2,0);   /* DUTY CYCLE */
            if (dcm==3) analogWrite(dcm3_pin2,0);   /* DUTY CYCLE */
            if (dcm==4) analogWrite(dcm4_pin2,0);   /* DUTY CYCLE */
          } else if (mode==1) {//L297
            if (dcm==1) digitalWrite(dcm1_pin2, LOW);  /* DIRECTION */
            if (dcm==2) digitalWrite(dcm2_pin2, LOW);  /* DIRECTION */
            if (dcm==3) digitalWrite(dcm3_pin2, LOW);  /* DIRECTION */
            if (dcm==4) digitalWrite(dcm4_pin2, LOW);  /* DIRECTION */
          }
          Serial.print("OK"); // tell Scilab that motor s initialization finished
                          // Cette commande sert à rien dans la toolbox de base,
                          // sauf si on prévoit d'ajouter des actions à l'init des moteurs
                          // par exemple chercher la position d'origine !
        }
       }
     }
  }
  val=-1;

}

//case M -> DC motor
else if(val==77){
    while (Serial.available()==0) {};
    val = Serial.read();
    /* the second received value indicates the motor number
       from abs('1')=49, motor1, to abs('4')=52, motor4    */
    if (val>48 && val<53) {
      dcm=val-48;               /* calculate motor number */
      while (Serial.available()==0) {};       // Waiting char
      val = Serial.read();
```

```
if (val==48 || val ==49){
   int direction=val-48;
   while (Serial.available()==0) {};       // Waiting char
   val = Serial.read();                    //reading next value = 0..255
   if (dcm==1){
    if(dcm1_mode==0){//L293
     if(direction==1){
       analogWrite(dcm1_pin1,val);
       analogWrite(dcm1_pin2,0);
     } else {
       analogWrite(dcm1_pin2,val);
       analogWrite(dcm1_pin1,0);
     }
    } else {//L298
     if (direction==0) digitalWrite(dcm1_pin2,LOW);
     if (direction==1) digitalWrite(dcm1_pin2,HIGH);
     analogWrite(dcm1_pin1,val);
    }
   }
   if (dcm==2){
    if(dcm2_mode==0){//L293
     if(direction==1){
       analogWrite(dcm2_pin1,val);
       analogWrite(dcm2_pin2,0);
     } else {
       analogWrite(dcm2_pin2,val);
       analogWrite(dcm2_pin1,0);
     }
    } else {//L298
     if (direction==0) digitalWrite(dcm2_pin2,LOW);
     if (direction==1) digitalWrite(dcm2_pin2,HIGH);
     analogWrite(dcm2_pin1,val);
    }
   }
   if (dcm==3){
    if(dcm3_mode==0){//L293
     if(direction==1){
       analogWrite(dcm3_pin1,val);
       analogWrite(dcm3_pin2,0);
     } else {
```

```
      analogWrite(dcm3_pin2,val);
      analogWrite(dcm3_pin1,0);
     }
    } else {//L298
     if (direction==0) digitalWrite(dcm3_pin2,LOW);
     if (direction==1) digitalWrite(dcm3_pin2,HIGH);
     analogWrite(dcm3_pin1,val);
    }
   }
   if (dcm==4){
    if(dcm4_mode==0){//L293
     if(direction==1){
      analogWrite(dcm4_pin1,val);
      analogWrite(dcm4_pin2,0);
     } else {
      analogWrite(dcm4_pin2,val);
      analogWrite(dcm4_pin1,0);
     }
    } else {//L298
     if (direction==0) digitalWrite(dcm4_pin2,LOW);
     if (direction==1) digitalWrite(dcm4_pin2,HIGH);
     analogWrite(dcm4_pin1,val);
    }
   }
}
if (val==114){//release motor
  if(dcm==1) {
  analogWrite(dcm1_pin1,0);
  if(dcm1_mode==0)  analogWrite(dcm1_pin2,0);
   }
 if(dcm==2) {
  analogWrite(dcm2_pin1,0);
  if(dcm2_mode==0)  analogWrite(dcm2_pin2,0);
 }
 if(dcm==3) {
  analogWrite(dcm3_pin1,0);
  if(dcm3_mode==0)  analogWrite(dcm3_pin2,0);
 }
 if(dcm==4) {
  analogWrite(dcm4_pin1,0);
```

```
          if(dcm4_mode==0)  analogWrite(dcm4_pin2,0);
         }
        }

     }
   val=-1;

  }


  //case R -> Analog reference
  if(val==82){
    while (Serial.available()==0) {};
    val = Serial.read();
    if (val==48) analogReference(DEFAULT);
    if (val==49) analogReference(INTERNAL);
    if (val==50) analogReference(EXTERNAL);
    if (val==51) Serial.print("v3");
    val=-1;
  }

} /* end loop statement                                */


/*************************************/
// Generic interrupt encoder functions//
/*************************************/
//Encoder on INT0
void encoder_0_change_m1() { //encoder0 mode 1x
  int chB=digitalRead(encoder_0_int2);
  if (!chB) { encoder_0_position++;}
  else { encoder_0_position--; }
}
void encoder_0_change_m2() { //encoder0 mode 2x
  int chB=digitalRead(encoder_0_int2);
  int chA=digitalRead(corresp[0]);
  if ((chA & !chB)|(!chA & chB)) { encoder_0_position++; }
  else { encoder_0_position--; }
}
void encoder_change_m4_A0(){//encoder0 mode 4x chA
```

```
  int chA=digitalRead(corresp[0]);
  int chB=digitalRead(encoder_0_int2);
  if ((chA & !chB)|(!chA & chB)) { encoder_0_position++; }
  else { encoder_0_position--; }
}
void encoder_change_m4_B0(){//encoder0 mode 4x chB
  int chA=digitalRead(corresp[0]);
  int chB=digitalRead(encoder_0_int2);
  if ((!chA & !chB)|(chA & chB)) { encoder_0_position++; }
  else { encoder_0_position--; }
}
//Encoder on INT1
void encoder_1_change_m1() { //encoder1 mode 1x
  int chB=digitalRead(encoder_1_int2);
  if (!chB) { encoder_1_position++;}
  else { encoder_1_position--; }
}
void encoder_1_change_m2() { //encoder1 mode 2x
  int chB=digitalRead(encoder_1_int2);
  int chA=digitalRead(corresp[1]);
  if ((chA & !chB)|(!chA & chB)) { encoder_1_position++; }
  else { encoder_1_position--; }
}
void encoder_change_m4_A1(){//encoder1 mode 4x chA
  int chA=digitalRead(corresp[1]);
  int chB=digitalRead(encoder_1_int2);
  if ((chA & !chB)|(!chA & chB)) { encoder_1_position++; }
  else { encoder_1_position--; }
}
void encoder_change_m4_B1(){//encoder1 mode 4x chB
  int chA=digitalRead(corresp[1]);
  int chB=digitalRead(encoder_1_int2);
  if ((!chA & !chB)|(chA & chB)) { encoder_1_position++; }
  else { encoder_1_position--; }
}
//Encoder on INT2
void encoder_2_change_m1() { //encoder2 mode 1x
  int chB=digitalRead(encoder_2_int2);
  if (!chB) { encoder_2_position++;}
  else { encoder_2_position--; }
```

```
}
void encoder_2_change_m2() { //encoder2 mode 2x
  int chB=digitalRead(encoder_2_int2);
  int chA=digitalRead(corresp[2]);
  if ((chA & !chB)|(!chA & chB)) { encoder_2_position++; }
  else { encoder_2_position--; }
}
void encoder_change_m4_A2(){//encoder2 mode 4x chA
  int chA=digitalRead(corresp[2]);
  int chB=digitalRead(encoder_2_int2);
  if ((chA & !chB)|(!chA & chB)) { encoder_2_position++; }
  else { encoder_2_position--; }
}
void encoder_change_m4_B2(){//encoder2 mode 4x chB
  int chA=digitalRead(corresp[2]);
  int chB=digitalRead(encoder_2_int2);
  if ((!chA & !chB)|(chA & chB)) { encoder_2_position++; }
  else { encoder_2_position--; }
}
//Encoder on INT3
void encoder_3_change_m1() { //encoder3 mode 1x
  int chB=digitalRead(encoder_3_int2);
  if (!chB) { encoder_3_position++;}
  else { encoder_3_position--; }
}
void encoder_3_change_m2() { //encoder3 mode 2x
  int chB=digitalRead(encoder_3_int2);
  int chA=digitalRead(corresp[3]);
  if ((chA & !chB)|(!chA & chB)) { encoder_3_position++; }
  else { encoder_3_position--; }
}
void encoder_change_m4_A3(){//encoder3 mode 4x chA
  int chA=digitalRead(corresp[3]);
  int chB=digitalRead(encoder_3_int2);
  if ((chA & !chB)|(!chA & chB)) { encoder_3_position++; }
  else { encoder_3_position--; }
}
void encoder_change_m4_B3(){//encoder3 mode 4x chB
  int chA=digitalRead(corresp[3]);
  int chB=digitalRead(encoder_3_int2);
```

```
  if ((!chA & !chB)|(chA & chB)) { encoder_3_position++; }
  else { encoder_3_position--; }
}
//Encoder on INT4
void encoder_4_change_m1() { //encoder4 mode 1x
  int chB=digitalRead(encoder_4_int2);
  if (!chB) { encoder_4_position++;}
  else { encoder_4_position--; }
}
void encoder_4_change_m2() { //encoder4 mode 2x
  int chB=digitalRead(encoder_4_int2);
  int chA=digitalRead(corresp[4]);
  if ((chA & !chB)|(!chA & chB)) { encoder_4_position++; }
  else { encoder_4_position--; }
}
void encoder_change_m4_A4(){//encoder4 mode 4x chA
  int chA=digitalRead(corresp[4]);
  int chB=digitalRead(encoder_4_int2);
  if ((chA & !chB)|(!chA & chB)) { encoder_4_position++; }
  else { encoder_4_position--; }
}
void encoder_change_m4_B4(){//encoder4 mode 4x chB
  int chA=digitalRead(corresp[4]);
  int chB=digitalRead(encoder_4_int2);
  if ((!chA & !chB)|(chA & chB)) { encoder_4_position++; }
  else { encoder_4_position--; }
}
//Encoder on INT5
void encoder_5_change_m1() { //encoder5 mode 1x
  int chB=digitalRead(encoder_5_int2);
  if (!chB) { encoder_5_position++;}
  else { encoder_5_position--; }
}
void encoder_5_change_m2() { //encoder5 mode 2x
  int chB=digitalRead(encoder_5_int2);
  int chA=digitalRead(corresp[5]);
  if ((chA & !chB)|(!chA & chB)) { encoder_5_position++; }
  else { encoder_5_position--; }
}
void encoder_change_m4_A5(){//encoder5 mode 4x chA
```

```
  int chA=digitalRead(corresp[5]);
  int chB=digitalRead(encoder_5_int2);
  if ((chA & !chB)|(!chA & chB)) { encoder_5_position++; }
  else { encoder_5_position--; }
}
void encoder_change_m4_B5(){//encoder5 mode 4x chB
  int chA=digitalRead(corresp[5]);
  int chB=digitalRead(encoder_5_int2);
  if ((!chA & !chB)|(chA & chB)) { encoder_5_position++; }
  else { encoder_5_position--; }
}

/**************************************
// Generic interrupt counter functions//
**************************************/
//Counter on INT0
void counter_0_change() { //counter 0
  counter_0++;
}
//Counter on INT1
void counter_1_change() { //counter 1
  counter_1++;
}
//Counter on INT2
void counter_2_change() { //counter 2
  counter_2++;
}
//Counter on INT3
void counter_3_change() { //counter 3
  counter_3++;
}
//Counter on INT4
void counter_4_change() { //counter 4
  counter_4++;
}
//Counter on INT5
void counter_5_change() { //counter 5
  counter_5++;
}
```

APPENDIX-B

Robotic Arm For Battery Cell Insertion

Extended Title: Inserting the cells of battery using Robotic Arm.
Brief Description : Traditional way of inserting cells of the battery in the container causes the folding of AGM shield which puts the pressure on the plates of the cells and in turn decreases the efficiency of the battery. Our aim is to design a robotic arm using sliding mode controller which will take cells one by one from the rack and insert them in the battery case without causing the AGM to fold and damage to the cells and increases the efficiency.
Scope of Work : Making conveyor belt, the Robotic arm and development of Sliding mode controller for automation of the industry.
Academic Objectives : This project will describe • Different features of sliding mode controller. • Designing of Conveyor belt • Designing of Robotic arm.
Application Objectives : *It can give the battery manufacturers the ability to place the cell inside the container without causing damage to the plates due to folding and hence increases the efficiency of battery.*
Previous Work Done on The Subject : • PID controlled conveyor belts. • PID controlled robotic arms.
Material Resources Required : Our project requires PRO-E, Matlab, SCI-lab, computer and controller.
No of Students Required : 4
Special Skills Required : • Detailed knowledge of sliding mode controller. • Knowledge of Matlab, Proteus and Scilab. G

Approval Status

Supervisor Name & Signature: _____

Assigned to:-

NC Sheharyar _____, NC Naveed Ramzan _____

NC Hafiz Muhammad Usman _____ PC M.Moin-ul-Haq _____.

HoD Signature _____

R&D SC Record Status

File # _____ **Coordinator Signature** _____.

APPENDIX-C

COST BREAKDOWN

Index	Equipment	Quantity	Unit price	Cost
1.	Motors(60 RPM)	3	4000	12000
2.	Motors(150 RPM)	2	1000	2000
3.	Arduino(mega)	1	3400	3400
4.	Fibre pcb	2	450	900
5.	Relays(12 V)	8	35	280
6.	Resistors	13	1	13
7.	Transistors(2N3904)	8	5	40
8.	Diodes(4007)	8	1	8
9.	Terminal Block	13	10	130
10.	Limit Switches	8	100	800
11.	Robotic Arm(material)	1	2500	2500
12.	Conveyor belt(material)	2	1000	2000
13.	Making	1	2000	2000
14.	Miscellaneous	1	4000	4000
15.	Printing	1	3000	3000
	Total Cost			33071