MINISTRY OF EDUCATION AND TRAINING

FPT UNIVERSITY

CAPSTONE PROJECT DOCUMENT

PROPELLER LED

Fast5		
Group Member	Nguyễn Thanh Tùng Đinh Quang Hải Đỗ Văn Ban Đinh Xuân Bách Nguyễn Thế Long	SE01960 SE02354 01870 60343 SE02241
Supervisor	Mr. Phan Duy Hùng	
Project Code	pLED	

Hanoi, April 10, 2014

Content

Content of Figures

ACKNOWLEDGEMENTS

During the implementation of this project, we have received overwhelming support from a number of people, to whom we would like to express our great appreciation. Without them, this project would have been done with much more obstacles than it was.

First and foremost, we would like to show our thankfulness to our supervisor of this project – Mr. Phan Duy Hùng. He has given us not only many valuable advices but also enthusiastic inspirations that helped raising our team members' spirit.

We would like to offer our special thanks to Đinh Xuân Bách, for his very big support so we can have a wide and convenient room to work.

Last but not least, we are thankful to our families and our friends for their constant encouragement and support throughout this project.

DICTIONARY

- **LED**: Light emitting diodes.
- **POV**: Persistence of Vision: is the phenomenon of the eye by which an afterimage is thought to persist for approximately one twenty-fifth of a second on the retina, and believed to be explanation for motion perception, however it only explains why the black spaces that come between each "real" movie frame are not perceived. The true reason for motion perception is the phi phenomenon.
- **Phi phenomenon**: the optical illusion of perceiving continuous motion between separate objects viewed rapidly in succession.
- **pLED**: Project code, Abbreviation for propeller display LEDs.
- **Propeller LED**: This is a linear array of light emitting diodes, rotating at a high angular velocity to generate a circular motion. By synchronizing these light emitting diodes to exploit concepts of persistence of vision, the product can create an LED screen from a straight array LEDs.
- **Perfboard:** A material for prototyping electronic circuits also called (DOT PCB).
- **RPM**: Rounds per minute
- **RF**: Radio Frequency
- **Hall sensor**: A component that can catch a magnetic signal.
- **Shift Register:** is a cascade of flip flops, sharing the same clock, in which the output of each flip-flop is connected to the "data" input of the next flip-flop in the chain, resulting in a circuit that shifts by one position the "bit array" stored in it, shifting in the data present at its input and shifting out the last bit in the array, at each transition of the clock input.
- **ICSP**: In Circuit Serial Programming
- **RS-232:** A standard for serial communication transmission of data.
- **I²C**: Inter-Integrated Circuit: used for attaching low-speed peripherals to a motherboard, embedded system, cellphone, or other digital electronic devices.
- **Development kit:** Include testing board, board design software, emulation software, firmware IDE.
- **FU:** FPT University.
- **PM:** Project Manager.

- **QA:** Quality Assurance
- **STD:** System Test Document.
- **SDD:** System Design Description.
- **SRS:** System Requirements Specifications .
- **RMP:** Risk Management Plan.
- **PMP:** Project Management Plan.
- **SUM:** Software User's Manual.
- **Frame:** One rotation of the board create an image with resolution 16 x 60.

  This one image is called a frame. With speed of 20 – 30 round/second, there are 20 – 30 frames per second.
- **NODE:** One rotation cycle of propeller is divided by 60 positions. Each position is called NODE.
- **DC:** Direct Current.
- **Commutator:** is the moving part of a rotary electrical switch in certain types of electric motors or electrical generators that periodically reverses the current direction between the rotor and the external circuit.

CHAPTER 1: INTRODUCTION

1. Video Propeller Led

Following is a short video of the completed product. It is will illustrate how does the propeller look like and we can imagine how does it works easier.

http://goo.gl/hcvRvc

2. Idea

Propeller clocks are nothing new to anyone who has been into electronics for a while. They use an idea called POV, Persistence of Vision, which means that if something appears in the same spot consistently, at least 24 times per second, our brains think that it's permanently there when it really is not. TV's and Monitors use this method of display, so it's not as uncommon as you might think.

After research some forums and existed project but there is no public source and tutorial about multicolor propeller. We decided to build a 7-color propeller LED on our own to satisfy our curiosity and also as our capstone project.

3. Existed Product

3.1 Bob Blick Propeller Clock – The first project[1]

This is the first product of the propeller clock idea. After that, many common products have been recreate and improve not only in design but also in feature.

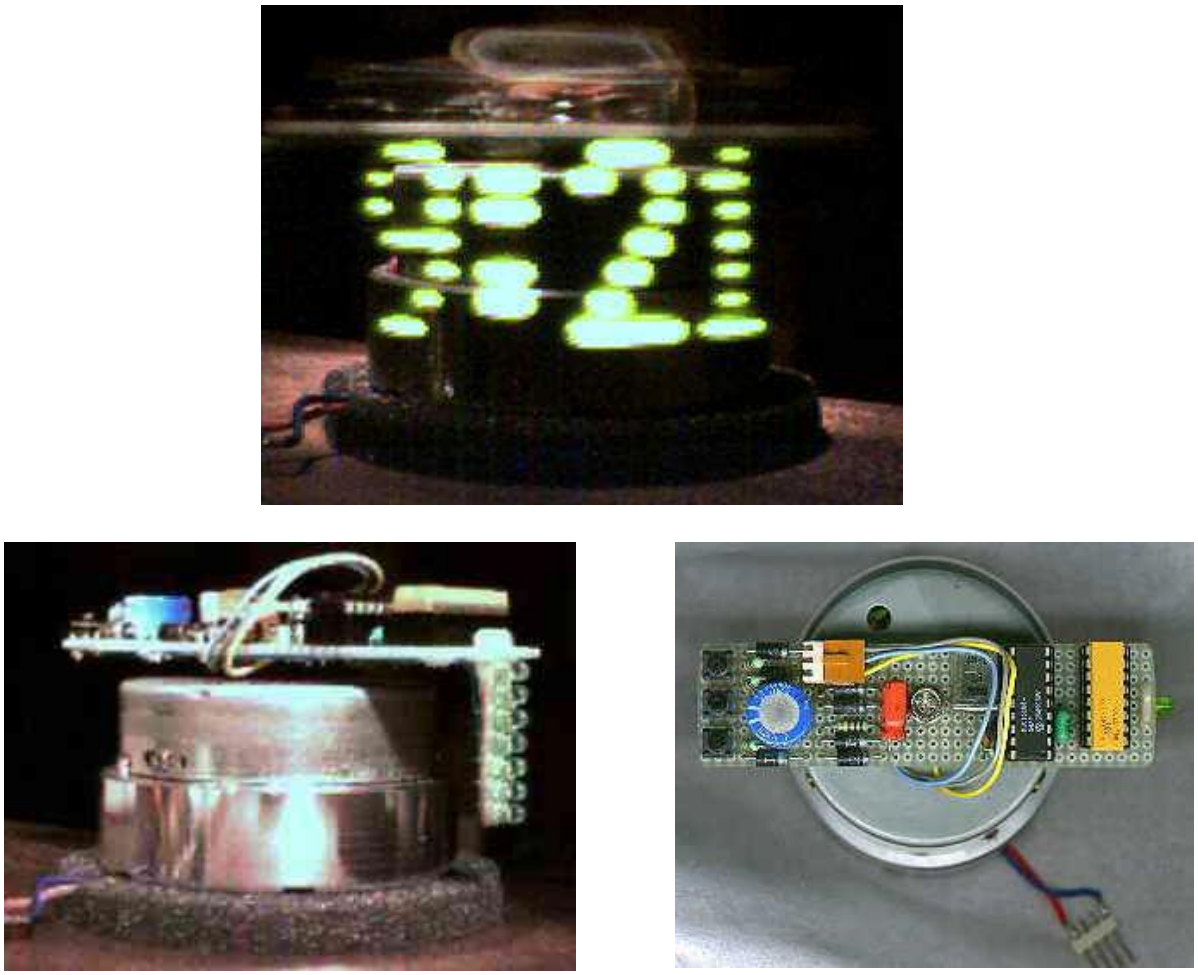


Figure 1: Bob Blick Propeller clock

The Propeller, a mechanically scanned LED clock by Bob Blick.

The clock is on a spinning piece of perfboard, but it must get power. They thought of many ways to do this, including using two motors(motor one has its shaft fixed to a base, and motor two spins the body of motor one, generating electricity), making a rotary transformer, or using slip rings.

They decided to do it another way, taking power from the spinning armature of a plain DC motor. In order to run the wires out of the motor, they removed the bearing from one end of the motor, leaving a big hole.

There are three terminals inside most small DC motors, and it acts a lot like three-phase alternating current, so it must be rectified back to DC. A nice side effect of this is that the position of the motor can be detected by taking one of the phases straight into the microprocessor.

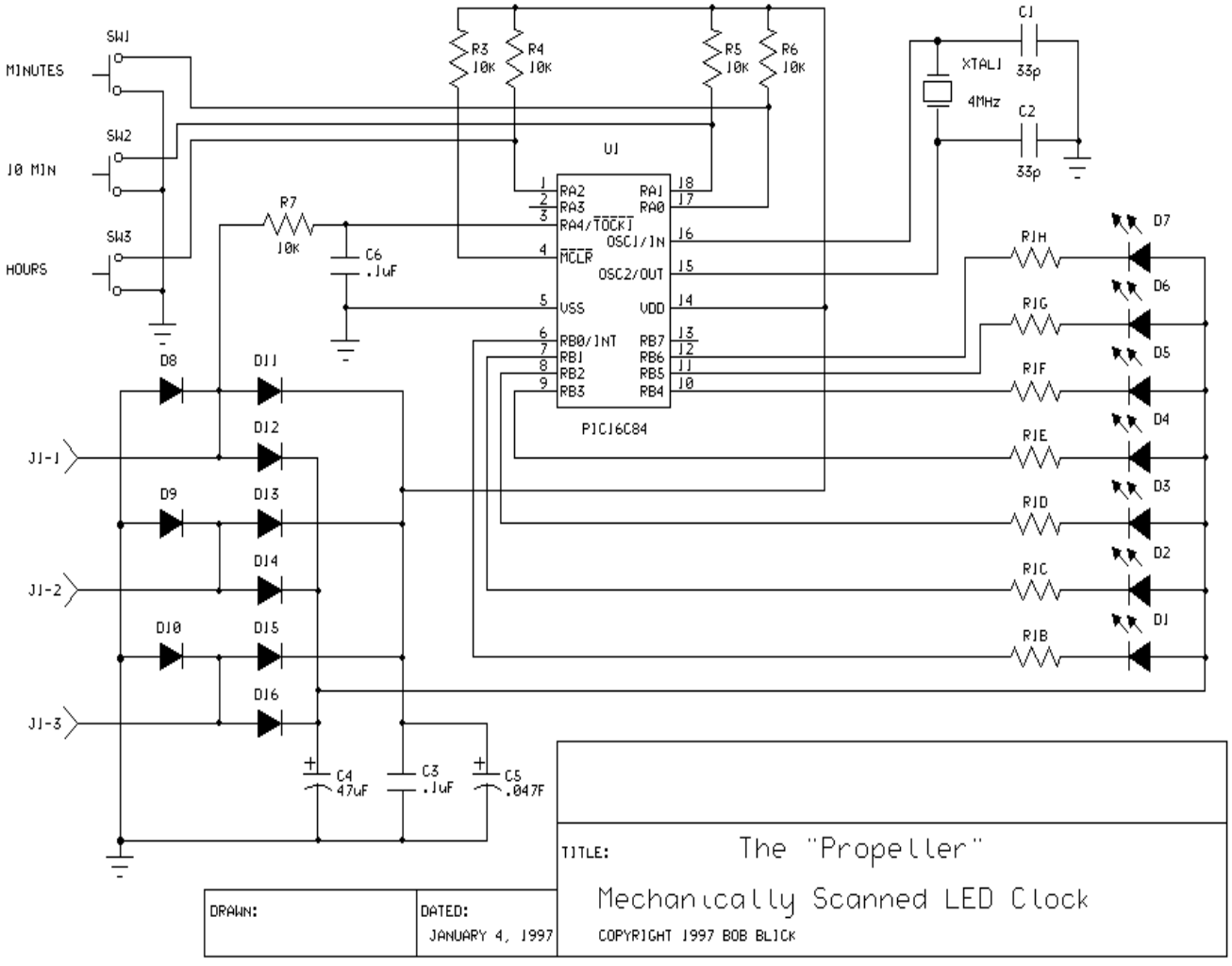


Figure 2: Schematic of Bob Blick's product

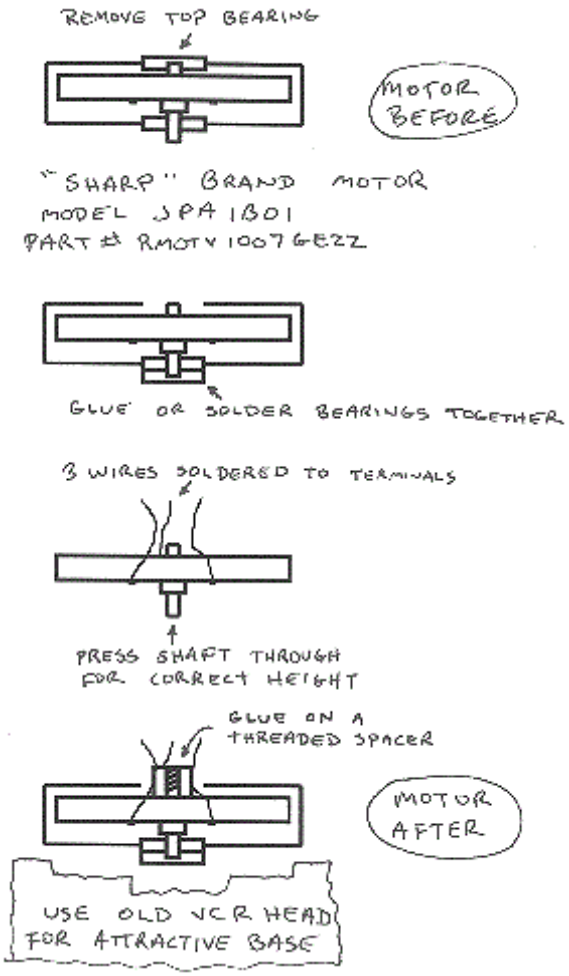


Figure 3: Motor design of BobBlick

3.2 AVClock[2]

Schematic Design:

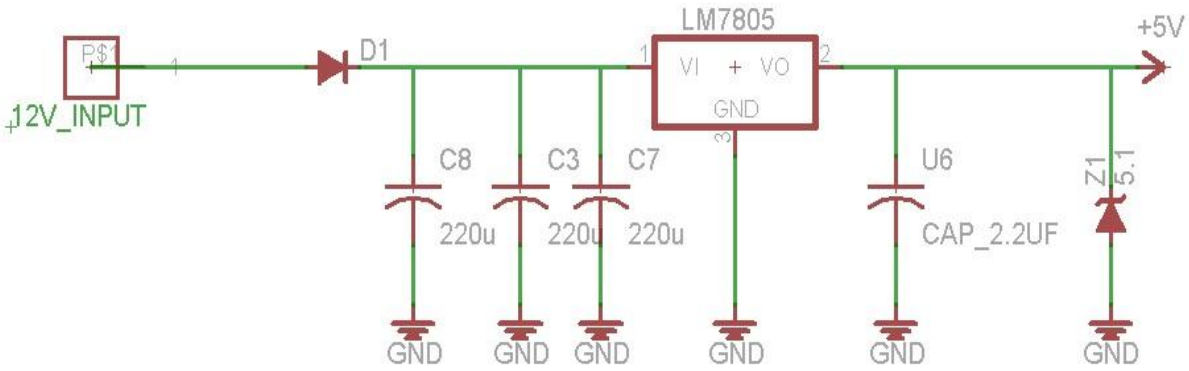


Figure 4: AVClock – Power source

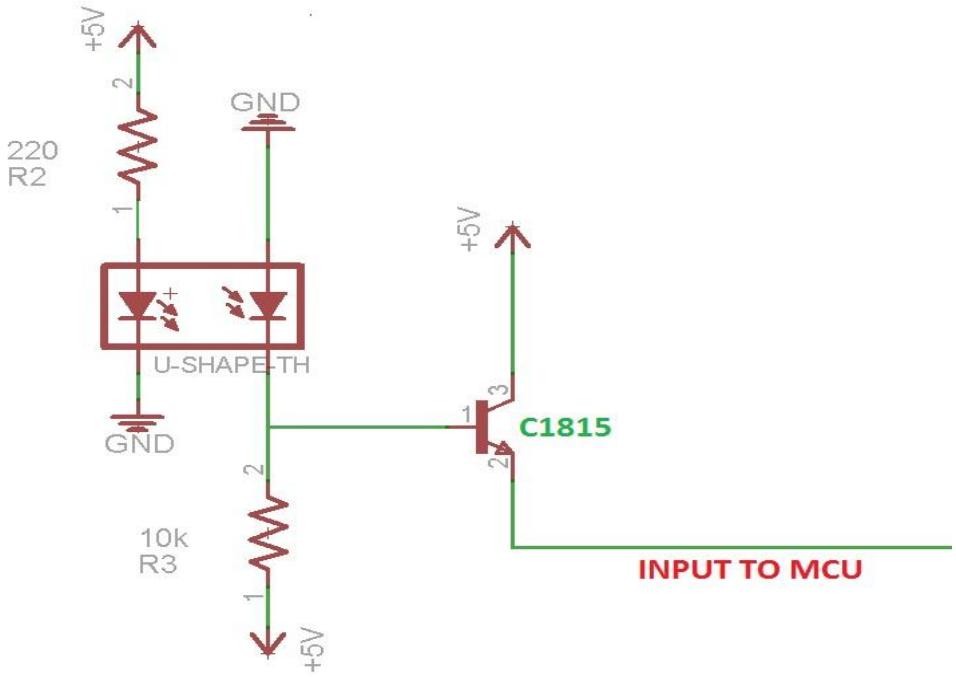


Figure 5: AVClock - External Interrupt

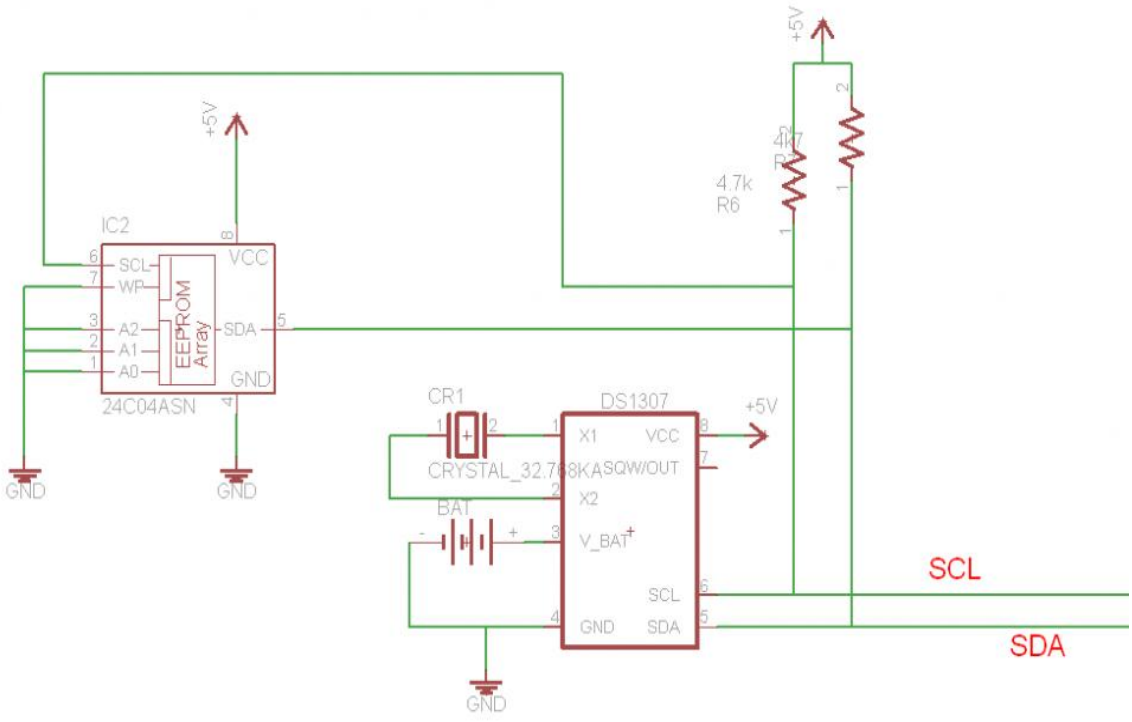


Figure 6: AVClock - I2C Communication

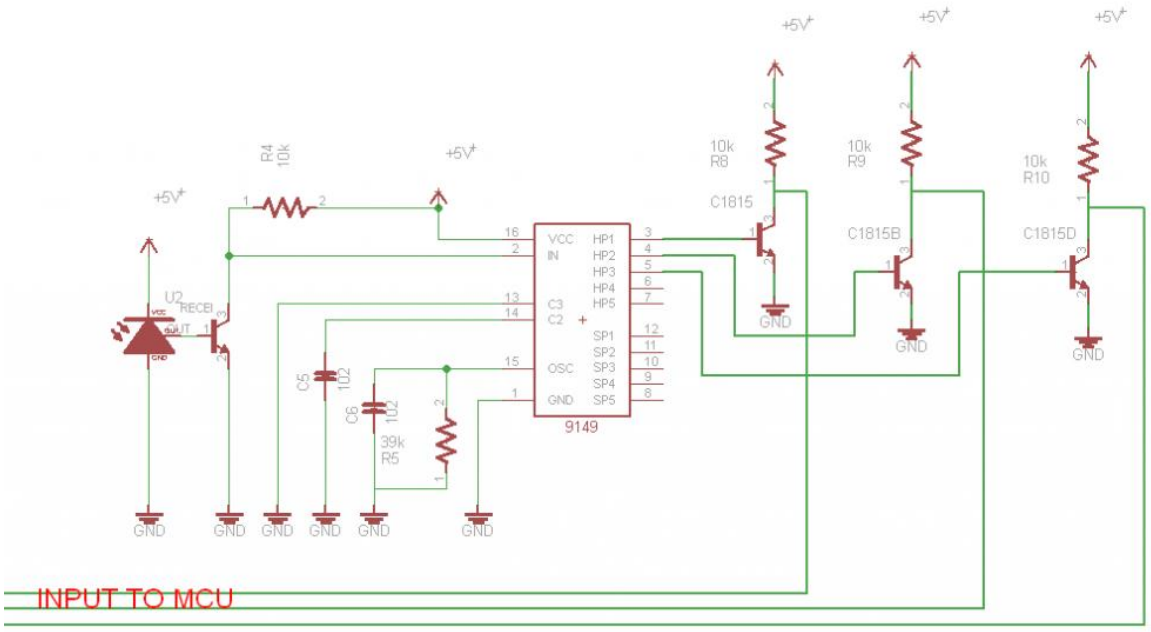


Figure 7: AVClock - Remote receiver

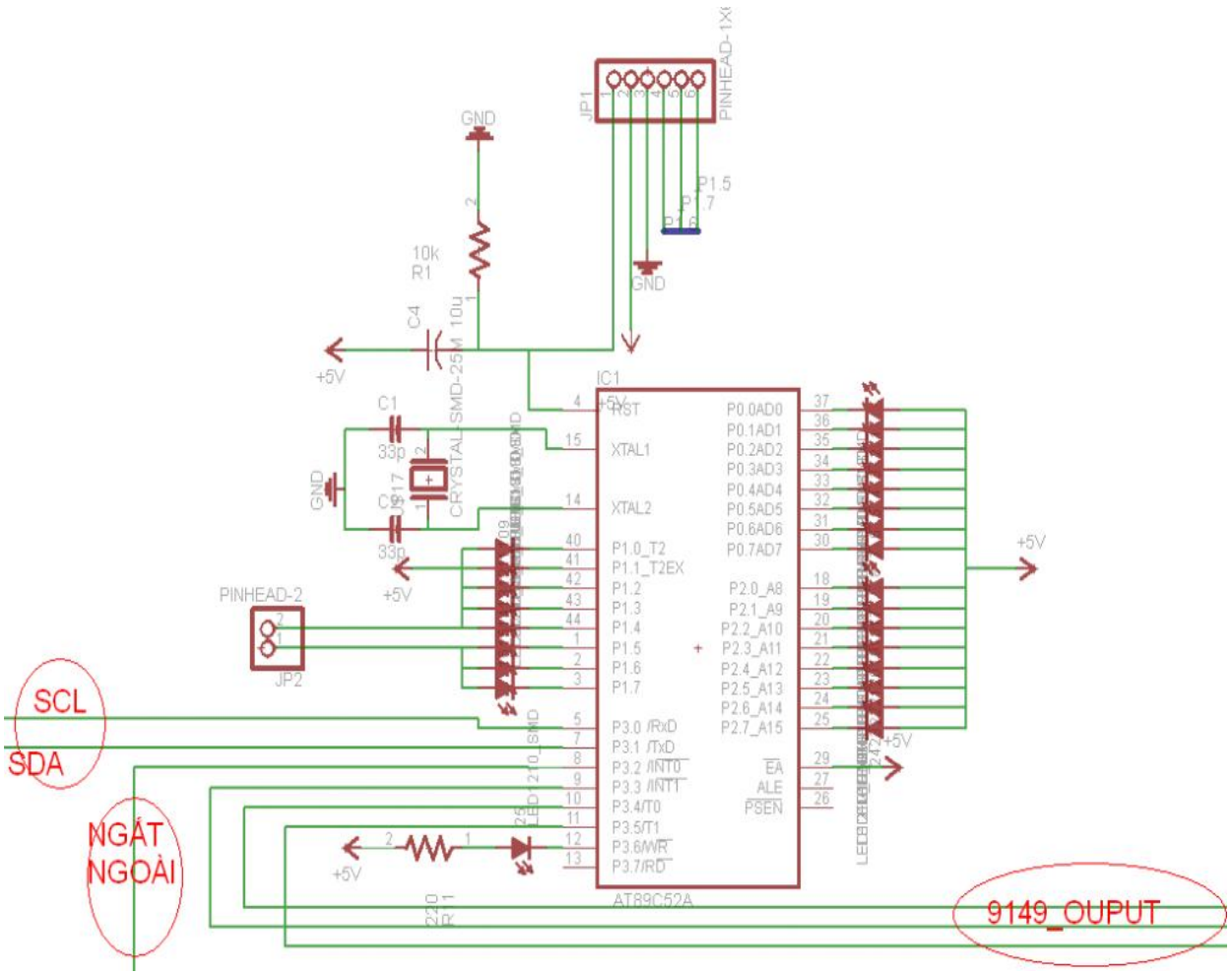


Figure 8: AVClock - Micro Controller

Final Product:

https://www.youtube.com/watch?v=yRS3Nvinm_I

Features:

- Display using single color LED but in high resolution (24 LEDs)
- Display Text, Clock, Animation
- Store real time
- Use infrared signal as remote system

4. Scope

The Project's purpose is researching and developing the Propeller LED that has following properties:

- ✓ Have 4 modes of display
- ✓ Run stable in hours
- ✓ Display real time exactly
- ✓ Store time and need only one time setup
- ✓ Remote by RF transmitter

CHAPTER 2: PROJECT MANAGEMENT

1. System process

Project's development process base on iterative and incremental development (IID).

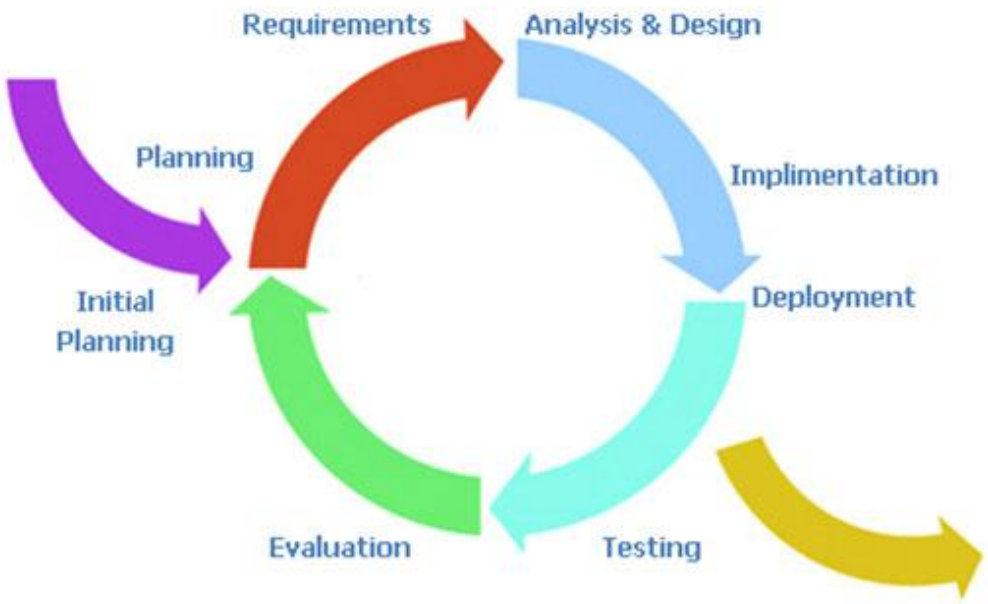


Figure 9: IID model

- IID provides regular feedback by breaking the project into iterations that are generally one to four weeks in length. The output of each iteration is working software with its necessary hardware or improvement on Schematic and Circuit board.

- The team estimates, plans and organizes work into iteration-sized chunks. The schedule feedback loop provides data that the team is on or off track by monitoring its ability to complete each iteration as planned.

- Requirements are the broken into smaller demonstrable units called stories. They are the estimate-able, testable, and deliverable units. They are small enough so can be completed within a single iteration.

2. Milestone

3. Roles and responsibilities

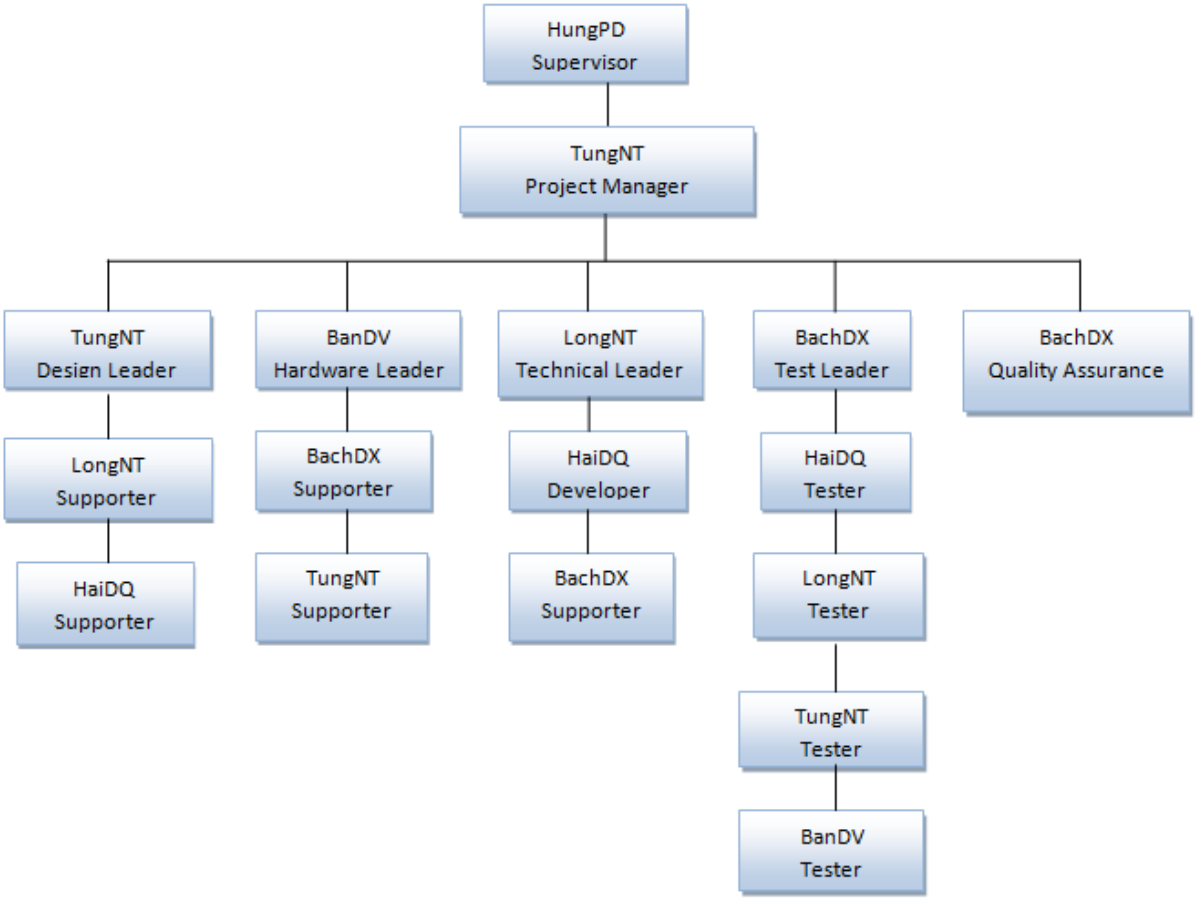


Figure 10: Team Roles

Full Name	Short Name	Phone	Email	Title
Phan Duy Hung	HungPD	0975 597 339	hungpd@fpt.edu.vn	Lecturer
Nguyen Thanh Tung	TungNT	0973 634 694	tungntse01960@fpt.edu.vn	Student
Do Van Ban	BanDV	01697 107 655	bandv01870@fpt.edu.vn	Student
Nguyen The Long	LongNT	0913 048 672	longntse02241@fpt.edu.vn	Student
Dinh Xuan Bach	BachDX	0985 866 690	bachdx60343@fpt.edu.vn	Student
Dinh Quang Hai	HaiDQ	0168 782 8721	haidqse02354@fpt.edu.vn	Student

Table 2.1: Contact's information

Process	Procedures	Responsibilities
Project Management	Project Planning	1. Prepares WBS: + Understands project scope, project context. Identifies project lifecycle + Prepares WBS 2. Estimates size of each work component in a measure suitable for the project, then gets it reviewed 3. Prepares schedule: + Defines detail activities with timelines (Gantt Chart) + Assigns the activities to project members 4. Prepares Project Management Plan (PMP)
Project Management	Project Executing	1. Conducts the meeting with team members. 2. Reviews timesheet and record actual resource utilization. 3. Modifies or Updates project plan If there are not any significant change on the plan we can modify it (without change request), if project have any major change, we must go back to Planning Project activities to update it.
Project Management	Project Monitor and Control	1. Monitor project against plan: 1.1. Monitor project parameters: + progress against schedule + resources provided and used + the attributes of work products and tasks + knowledge and skill of project team 1.2. Document the significant deviations in the project planning parameters 2. Project Control: + Analyze issues
Project Management	Project Closure	1. Prepare the evidence (work products) to demonstrate that phase had got its objectives 2. The PM prepares the Project Final Report. 3. Select and document lesson learn and best practices 4. Prepare project close presentation material 5. The PM ensures the project end backups are taken.
Risk Management	Identify, analyze risks and create avoidance plan	1. The PM and senior project team members meet to identify and analyze risks. The identified risks are documented 2. For each risk identified, agrees with the team on the probability and impact. Have plan to avoid risk. 3. Assigns members to performs the action according to mitigation/ contingency plan.
Configuration Management	Baseline	Create repository on *code.google.com* and grant permission to project's member

Table 2.2: PM's responsibilities

Procedures	Responsibilities
Low Level Design	Design propeller's display
Coding & Unit Testing	1. Using the Unit Specifications document & the coding guidelines, the designated developer writes/ modifies the code for the assigned unit. The developer may also need to refer to earlier documents, such as SRS. 2. Readies the code for execution. Typically, this involves compiling the unit code. If decided by the PM, the code is reviewed before unit testing. Typically, this is done for critical units. If a review is performed, review comments are incorporated in the code, and this is verified 3. Updates the Requirement traceability matrix 4. Tests the unit using white-box unit test cases. Defects are logged in the defect log. The unit is debugged by the developer and the testing repeated till the unit is tested OK

Table 2.3: Developer's responsibilities

Procedures	Responsibilities
High Level Design	1. Prepares the Integration and System test case. 2. Update Requirement Traceability Matrix
Low Level Design	Prepares black box component test cases
Integration, System Testing	1. Performs Integration test as per test plan and integration test cases 2. After Integration test, performs the system test as per test plan / system test cases

Table 2.4: Tester's responsibilities

Process	Procedures	Responsibilities
Project Management	Project Planning	1. Assist in project planning activities + Life cycle selection + Establishing the project's defined process + Setting quality goals and other objectives + Understanding and usage of templates, forms, guidelines and checklists + Identifying process training needs (as applicable) + Getting information (like historical data, best practices, lessons learnt from similar projects, risks etc.) useful for project planning
Project Management	Project Executing	Assist in project execution activities + Re-planning + Facilitating team in understanding templates/forms + Timely highlighting of process related potential issues (Formal) + Participating in weekly or important meetings

Project Management	Project Closure	Assist in project closure activities + Providing projects experiences

Table 2.5: QA's responsibilities

4. Risk analysis and management

No	Risk	Analyze	Avoidance
HARDWARE			
1	Speed of motor is unstable	Power consumption of circuit board is inconstant	Separate power source of motor and circuit board
2	Base has strong vibration	Motor run at high speed creates vibration	Increase the weight of the base to restrict the vibration
3	Electric transmission is glimmering	Use carbon brush can make the current unstable because after run for a long time the brush become dull	Use rolling-element bearing to transmit the current
4	Propeller is imbalance	The distribution of components on the circuit board is not proportional	Attach some heavy components to the circuit board to balance it
5	Operation of IC is unstable	Some ICs are made in china, so the quality is not guaranteed	Test ICs carefully before use
6	High speed of propeller	Some components can fly out	Use glue to fix components steady attach to the board
7	Broken motor	Motor can be blown out after run for a long time	Make backup motors
SOFTWARE			
8	Micro chip is not fast enough	The program need faster chip to execute more command in a unit of time	Optimize code using better algorism. Modify CCSC's library to reduce the number of command
9	The amount of RAM is small	RAM is not enough to store all necessary data	Create algorism to locate data in ROM
10	I2C communication's speed is low	Each time read data from Real Time Clock, displaying LEDs get flicker	Only read time data after couples minutes

Table 2.6: Risk Management

5. Configuration Management

5.1 Project structure (SVN)

Repository				Guidelines
Branches				*Baselines*
Tags				
Trunks	**01. Pre-Sales**	**011. SOW & Proposal**		*Proposal docs*
		012. Contract		*Project Contracts*
	02. Project Management	**021. Initiation**		*Initiation Note, Project charter*
		022. Estimation		*Project Estimation Sheet*
		023. Project Plan		*Master plan, detail plan (.mpp file, excel)*
		024. Reporting	**Account status**	
			Timesheet	*Timesheet of team*
			MoM	*- Meeting minute* *- Email , chat notes with customer*
			PMR	*Report process to customer*
		025. Trackers	**Issue Tracker**	
			Project Tracker	
		026. Feedback	**Appreciation**	
			Customer Feedback	*Leakage list*
			Rewards and Recognition	
		027. Handover		*Handover document or log between project team members.*
		028. Client Sign-off		*Document signed-off by customers supply*
		029. Closure		*Project Closure meeting slide, Project Scoring, …*
	03. Engineering	**031. Guidelines and Standards**		*- Coding convention* *- Source code model of customer* *- Guideline of team, QA, PM*

		032. Requirement	Business Requirements	*- SRS, Scope Statement,* *- Q&A* *- File graphic design (GUI file)*
			Functional Requirements	
			Impact Analysis	
			RTM	
		033. Design	High level design	*Document design (Basic design)*
			Low level design	*Document design (Detail - Design)*
		034. Implementation		*Source Codes implemented by team*
		035. Peer Review Logs		
		036. Testing	Test Plan	*Test plan of tester leader/tester/PM*
			Test Case	*Test case / Test checklist of tester*
			Test Report	*Test report of Tester*
		037. Transition		*Source Codes implemented by team*
	04. Support	041. CM		*Document CM . Example: CM guidelines*
		042. DAR		
		043. MAA		
		044. PPQA		*Document PPQA: QA Plan, QA report, Tailoring*
	05. References	0.51. Tools		
		0.52. Others		*Document consult of team*
		0.53. Customer supply		*Document consult by customer supply, test case updated*
		0.54. Template		
		0.55. Training		*Document training*
	06. Temp			*Document personal*

Table 2.7: pLED_CM Guidelines

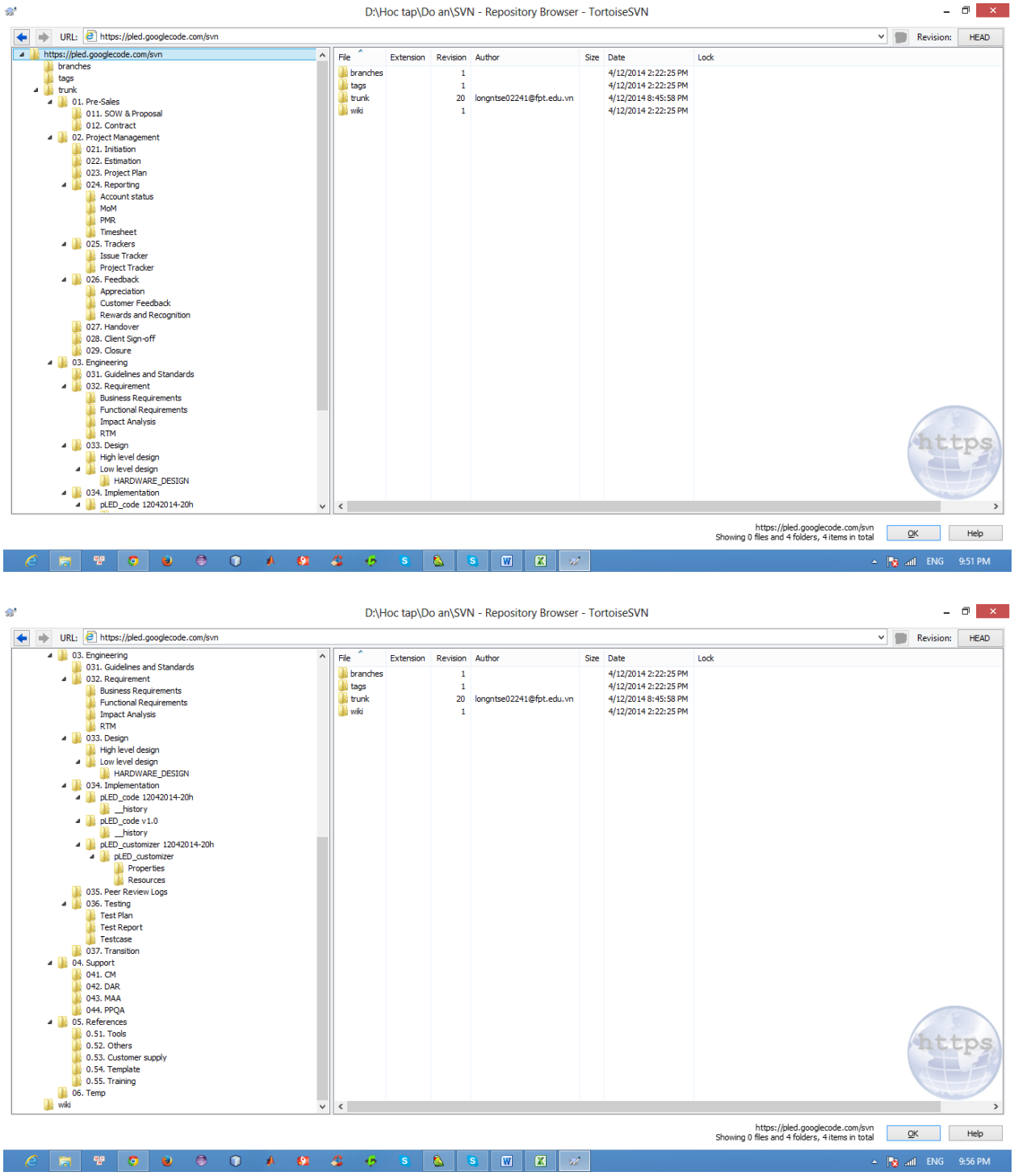


Figure 11: Project directory structure

Repository: https://pled.googlecode.com/svn/

5.2 Version Control

When change document must be set version following: m.n

Stage in Lifecycle	Version Number
Initial Version	0.00
BaseLine	1.0
Changes to Baseline Documents	1.1
Major Enhancements	2.0

For example: pLEDv1.0

5.3 Rules

- ✓ Document and file must be save on right folder SVN
- ✓ Version of document must be increase if it was changed
- ✓ Source code must be create baseline when release to customer and on time follow rules of project
- ✓ When commit file and document into SVN member must be write comment or message
- ✓ Bug, task, Q&A, Issue must be updated status and re-plan over due date daily

6. Communication and meeting

All Meeting minutes are noted down follow the below template and store at "trunk/02. Project Management\024. Reporting\MoM" directory of project.

Meeting minutes 1			
1/6/2014	9:00 AM	FU's Library	
Meeting	TungNT		
Type of	Face to face		
Facilitator			
Note taker	HaiDQ		
Timekeeper	TungNT		
Attendees	LongNT, BanDV, BachDX		
Idea Research			
30 mins	TungNT		
Discussion	Review existed product and collect requirement to create own product.		
Conclusions	Have first requirement document of project.		
Action Items		Person	Deadline
SRS		HaiDQ	1/6/2014

Table 2.8: Meeting minutes template

CHAPTER 3: SYSTEM REQUIREMENT

1. User Requirement Specification

1.1 Purpose

The purpose of first part of this chapter is to specify functional, nonfunctional requirements that the product must satisfy and some use cases that user can follow to use the product easier.

1.2 Overall Description

1.2.1 Business Process Overview

The pLED system main purpose is for entertainment and to demonstrate the ability of the PIC microcontroller in its various features. Also to prove the skills of development team in rebuild a device base on a common known physical phenomenon.

1.2.2 Product Features

The pLED is a mechanic and electronic system with the main feature is a series of LEDs that rotate around a focal point. With enough rotational speed, it will display a desire image on a circle plane. The mechanism is simplification of a TV screen.

1.2.3 User Characteristic

User of the system is any individual who wish to decorate their home or anyone who interest in physic, electronic device. The pLED can be used for demonstrate only or can be sold as a family commodity.

Normal user: can understand the main features of the product and can use it in no more than 15 minutes.

Advanced user: have technical knowledge about the product, configure it through computer software, and change its parameters.

1.3 Functional requirement

- Display real time clock or image on rotating board with series of LEDs. The control task is responsibility of a PIC controller. The board is rotated by a DC motor.
- User can interact with the system through a RF module.
- System can be configured through computer software.

1.4 Non-functional requirement

- The display quality of clock or image on the rotation board.
- The stability of the system, power supply, running time.
- Response time of the system, after power on, response of user input.

2. System Requirement Specification

2.1 Purpose

The software requirement specification assures the project management stakeholders and client that the development team has really understood the business requirements documentation properly. This also provides confidence that the team will develop the functionality which has been detailed.

This part of chapter 3 is documented in such a way that it breaks the deliverables into smaller components. The information is organized in such a way that the developers will not only understand the boundaries within which they need to work, but also what functionality needs to be developed and in what order.

2.2 System purpose

Propeller LED system is use to display real time and image or text.

2.3 Scope

- A functional Propeller LED system, display accurate real time and date right after plugged in a power source.
- A propeller display with battery to show an exact time/date whenever turned on.
- A set of display mode: Analog clock, digital clock, text/image, that can be change through a dedicated remote control.

- The dedicated remote using radio wave to control.

2.4 Overall Description

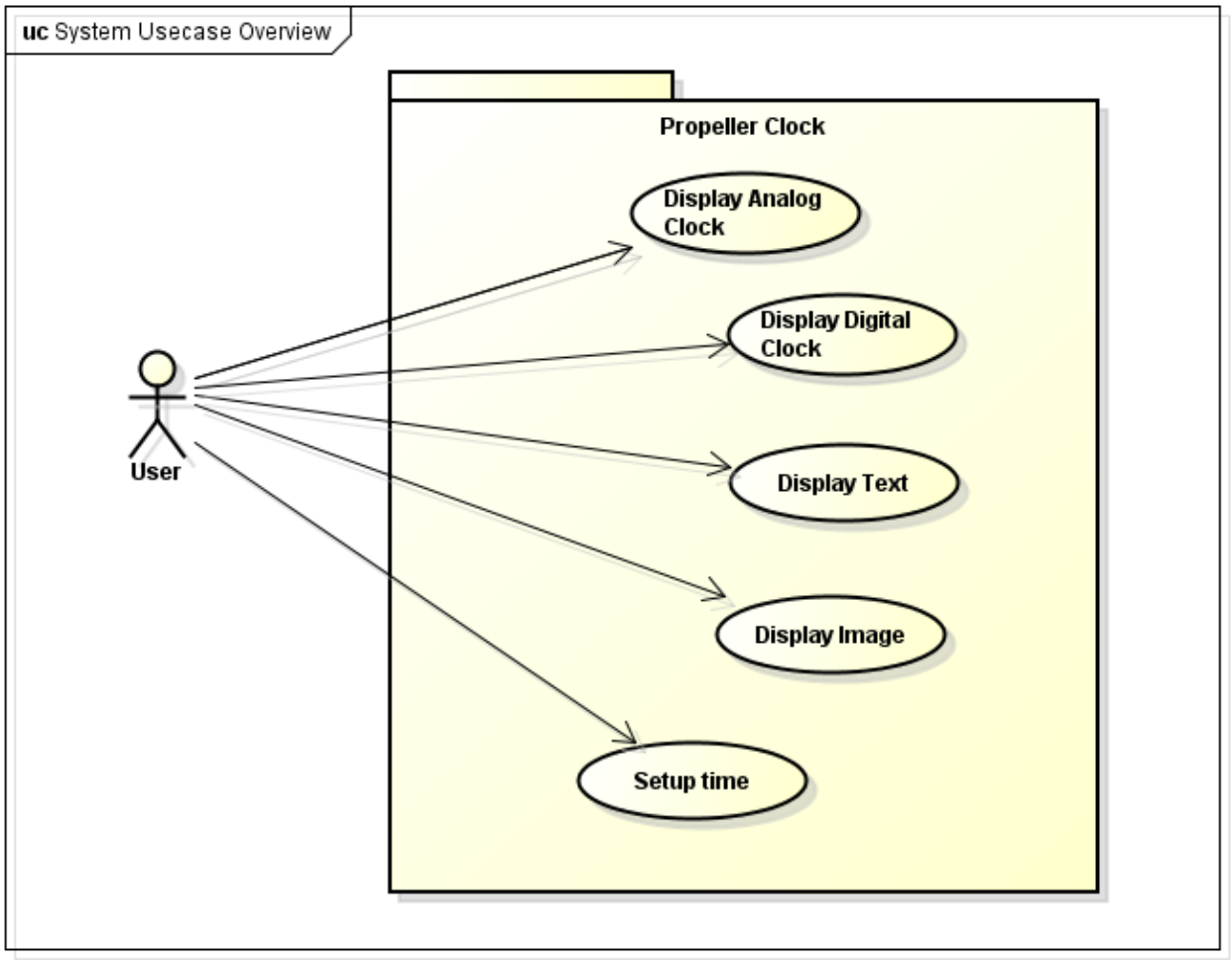
2.4.1 System Overview

- Propeller LED system rotates then displays image, text within selection from user remote by radio control.
- Propeller LED system will count real time when system is off.

2.4.2 Product Features

- Display clock in real-time.
- Two to three display option:
  - Digital clock
  - Analog clock
- Display image or text
- User can change display mode by remote control.
- Multiple color display

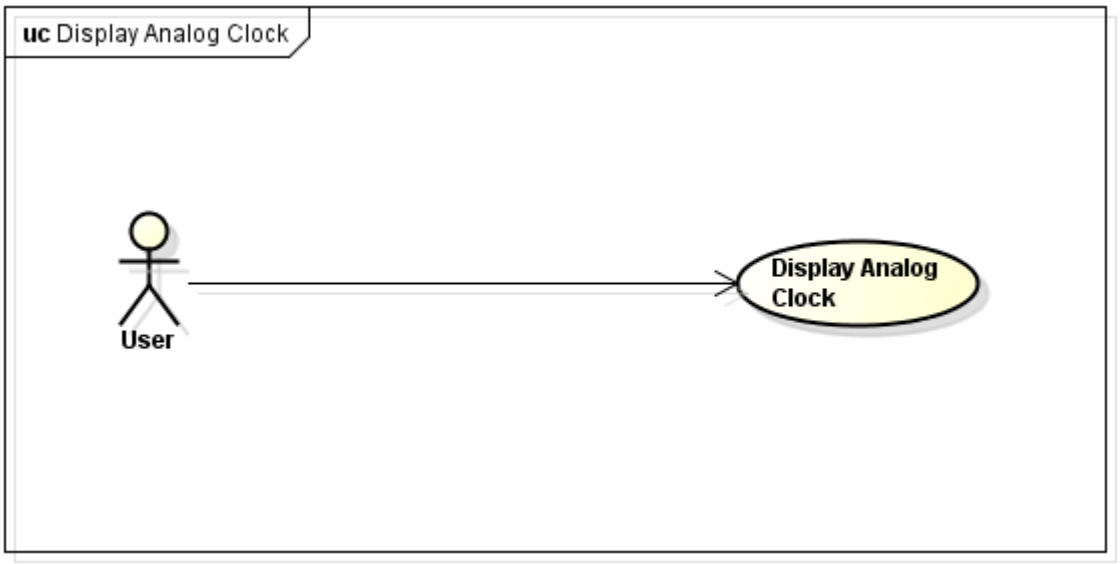
2.5 Functional specification



Five usecases: Display Analog Clock, Digital Clock, Text, Image and Setup Time

2.5.1 UC001: Display Analog Clock

a) Use Case Diagram

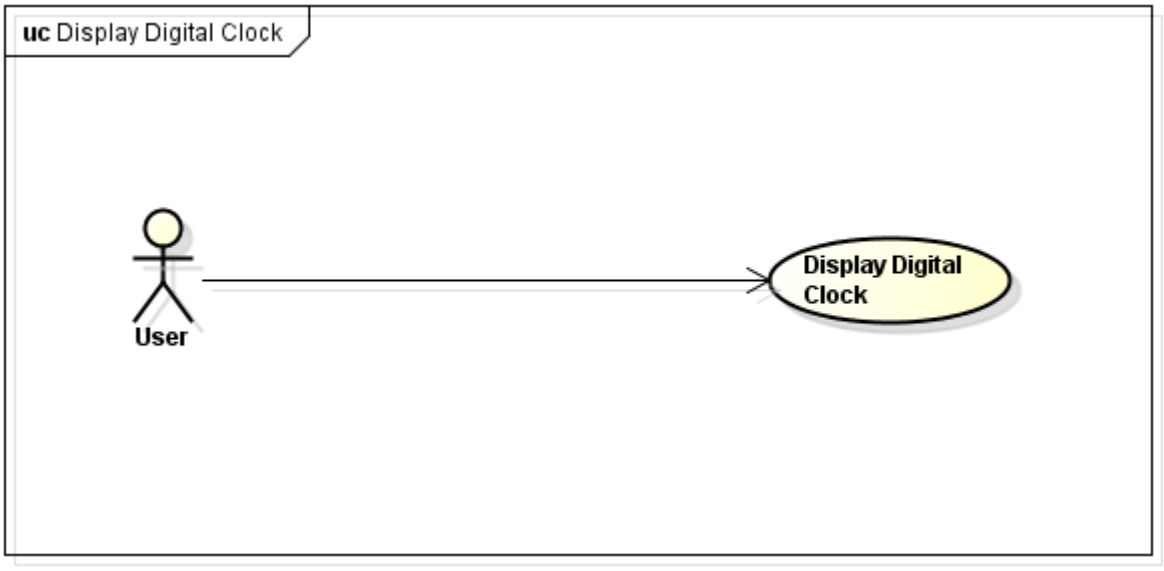


b) Use Case Specification

Use Case ID :	UC001			
Use Case Name :	Display Analog Clock			
Description :	User press button “A” on RF, then system will display analog clock.			
Primary Actor:	User			
Pre-Condition:	Power input 220V AC, Proper LED rotates			
Post-Condition:				
Flow of Events			Actor Input	System Respond
		1	Press button “A” on RF	
		2		System display analog clock

2.5.2 UC002: Display Digital Clock

a) Use Case Diagram



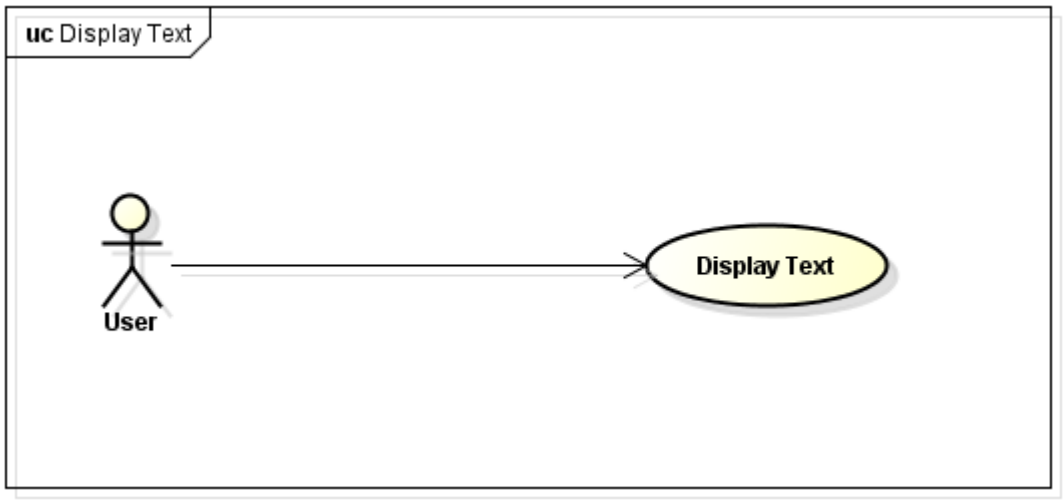
b) Use Case Specification

Use Case ID :	UC002
Use Case Name :	Display Digital Clock
Description :	User press button “B” on RF, then system will display digital clock.
Primary Actor:	User

Pre-Condition:	Power input 220V AC, Proper LED rotates				
s Flow of Events			Actor Input	System Respond	
		1	Press button “B” on RF		
		2		System display digital clock	

2.5.3 Display Text

a) Use Case Diagram

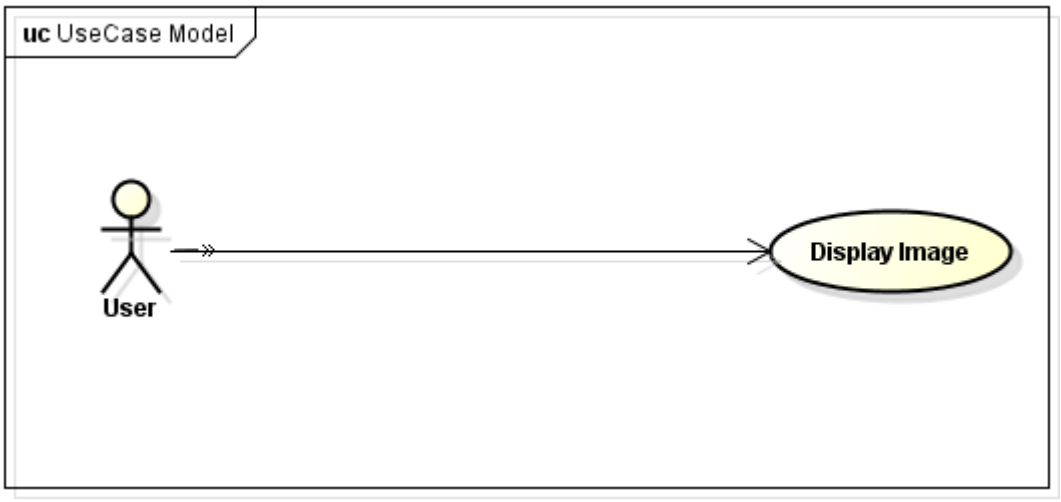


b) Use Case Specification

Use Case ID :	UC002				
Use Case Name :	Display Text				
Description :	User press button “C” on RF, then system will display text.				
Primary Actor:	User				
Pre-Condition:	Power input 220V AC, Proper LED rotates				
Post-Condition:					
Flow of Events			Actor Input	System Respond	
		1	Press button “C” on RF		
		2		System display text	

2.5.4 Display Image

a) Use Case Diagram

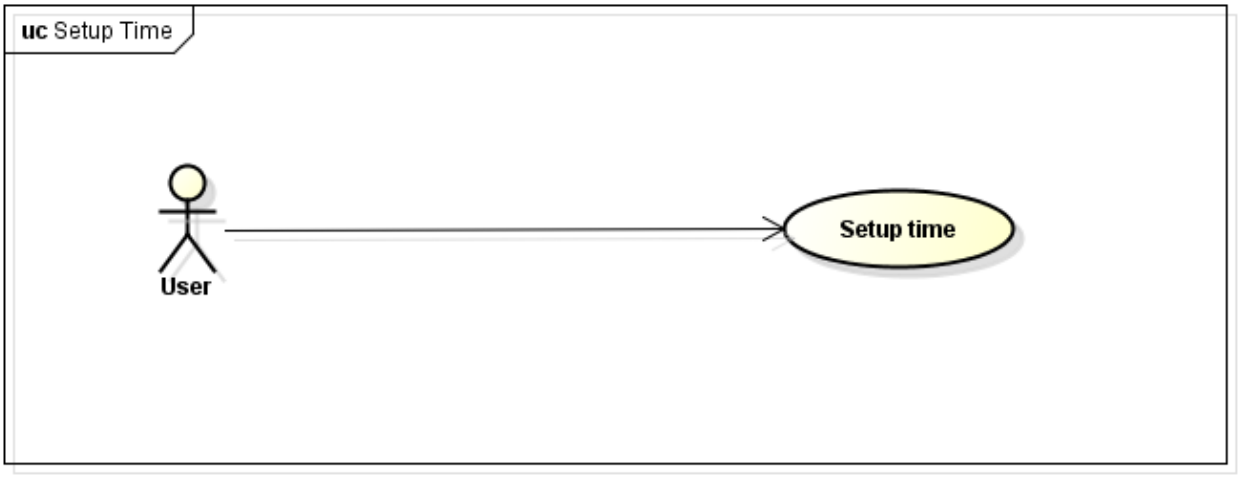


b) Use Case Specification

Use Case ID :	UC004				
Use Case Name :	Display Image				
Description :	User press button “D” on RF, then system will display image.				
Primary Actor:	User				
Pre-Condition:	Power input 220V AC, Proper LED rotates				
Post-Condition:					
Flow of Events			Actor Input	System Respond	
		1	Press button “C” on RF		
		2		System display image	

2.5.5 Setup Time

a) Use Case Diagram



b) Use Case Specification

Use Case ID :	UC005			
Use Case Name :	Setup Time			
Description :	User connect system with application. Setup – time and click OK button			
Primary Actor:	User			
Pre-Condition:	Proper LED not rotates. System off			
Post-Condition:				
Flow of Events			Actor Input	System Respond
		1	Connect system with application through RS232, click button “Connect”	
		2		Application display connect
		3	Setup time , then click button “Update”	

2.6 Non-functional specification

2.6.1 Hardware

- Power supply is robust.

- Motor rotate stable.
- Real time display.

2.6.2 Performance Requirements

- Speed of motor at least 20 cycle per second.
- Picture display smoothly.
- Ability to run for couple of hours.
- 220V AC Power supplier.
- Run at temperature from -10^0C to 70^0C.
- When motor run, power up, system display within 3s.
- When use remote control, system respond after 1s.

2.6.3 Design Constraints

- Speed of motor enough to display picture smoothly
- Design motor supply into board with enough power
- System vibration suitable.

2.6.4 Usability

User can use product after 5 minutes of reading guideline.

2.6.5 Security

Safety for user

2.6.6 Protection Requirements

Avoid electric shock – proof.

2.6.7 Hardware/Software Requirements

- Design power supply 12V DC
- Design board easy to mechanical welding
- Design PC software to connect board via RS232
- Using RF component to remote system

3. Infrastructure and Tools

3.1 Hardware

No1	Item	Why we use it?
1	PIC16F887[5]	Cheaper and we already used it
2	DS1307[6]	Useful to get real-time clock
3	EEPROM	Useful to save data
4	LED RGB	More color, more option
5	RF Module	Use radio wave is more convenient than infrared signal. This type of signal is more stable and cant be block by obstacles.
6	Shift Register[7]	Use to store data for LEDs because number of pin of Micro Controller is not enough to control LEDs directly
7	IC 7805	This IC is popular positive-voltage regulator.
8	20MHz Crystal	Max frequency range of Pic887.
9	33KHz Crystal	Required frequency of DS1307

Footprint	Comment	LibRef	Designator	Description	Quantity
BATTERY	Battery	Battery	Bat		1
CAP POL 2	Cap Pol3	Cap Pol3	C1, C2	Polarized Capacitor (Surface Mount)	2
SM/C_0805	Cap Semi	Cap Semi	C5, C7, C8	Capacitor (Semiconductor SIM Model)	3
DO-35	Diode 1N4148	Diode 1N4148	D1	High Conductance Fast Diode	1
HDR1X3	Header 3	Header 3	Hall1	Header, 3-Pin	1
HDR1X5H	Programming	Header 5H	J1	Header, 5-Pin, Right Angle	1
RGBLED5050	or 5050 LEDs, the order of the LEDs may vary from one manufacturer to another!	RGBLED5050	LED1, LED2, LED3, LED4, LED5, LED6, LED7, LED8, LED9, LED10, LED11, LED12, LED13, LED14, LED15, LED16		16
HDR1X2H	DC 12V	Header 2H	P1	Header, 2-Pin, Right Angle	1
HDR1X3	7805	Header 3	P2	Header, 3-Pin	1
HDR1X3H	Header 3H	Header 3H	P3	Header, 3-Pin, Right Angle	1
SM/C_0805	Res Semi	Res Semi	R1, R2, R3, R5, R6, R7, R8	Semiconductor Resistor	7
HDR1X7	RF Module	Header 7	RF1	Header, 7-Pin	1
8S1_N	AT24C01BN-SP25-T	AT24C01BN-SP25-T	U1	1Kbit (128 x 8), 2-wire Bus Automotive Serial EEPROM with Write Protect, 2.5V, 8-Pin SOIC, Tape & Reel	1
SO16	74595	74595	U2, U4, U5, U7, U8, U9	8-BIT SFT REG WIRTH OUTPUT LCH	6
SOIC8N_N	DS1307Z	DS1307Z	U3	64 X 8 Serial Real-Time Clock	1
TQFP-PT44_N	PIC16F887-E/PT	PIC16F887-E/PT	U6	Flash-Based, 8-Bit CMOS Microcontroller with nanoWatt Technology, 8K (x14-Bit words) Program Memory, 624 Bytes Data Memory, 35 I/O pins, 44-Pin TQFP, Standard VDD Range, Extended Temperature	1
CFPX-93	32,768KHz	CFPX-93	Y1	Surface Mount Quartz Crystal	1
CRYSTAL	20MHz	XTAL	Y2		1
					47

3.2 Software and tool

- CCS C 5.15 (IDE)
- ISIS Proteus (emulation)
- PICKIT2, ICD2 Programmer
- Electronic multi-meter
- DC motor
- Altium Designer 10
- Microsoft Visual Studio 2012

CHAPTER 4: SYSTEM DESIGN

1. Common Design

The system is controlled by interrupts[3]. This means that tasks performed by the system are triggered by different kinds of events; an interrupt could be generated, for example, by a timer in a predefined frequency, or by a serial port controller receiving a byte.

These kinds of systems are used if event handlers need low latency, and the event handlers are short and simple. Usually, these kinds of systems run a simple task in a main loop also, but this task is not very sensitive to unexpected delays.

Sometimes the interrupt handler will add longer tasks to a queue structure. Later, after the interrupt handler has finished, these tasks are executed by the main loop. This method brings the system close to a multitasking kernel with discrete processes.

1.1 Block Diagram: Hardware Interface

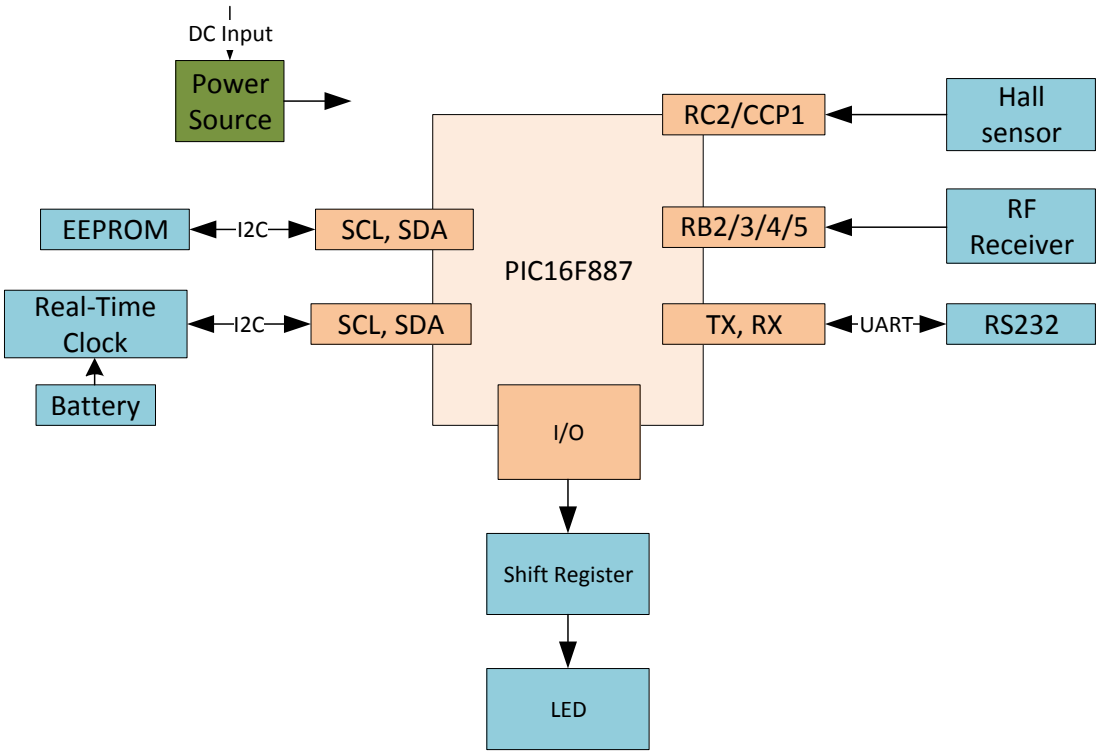


Figure 12: System Block Diagram

1.2 User Interface

- RF transmitter: use to change the display mode of Propeller within 15m.

Figure 13: RF transmitter

- The Propeller: Read-to-run dedicated.

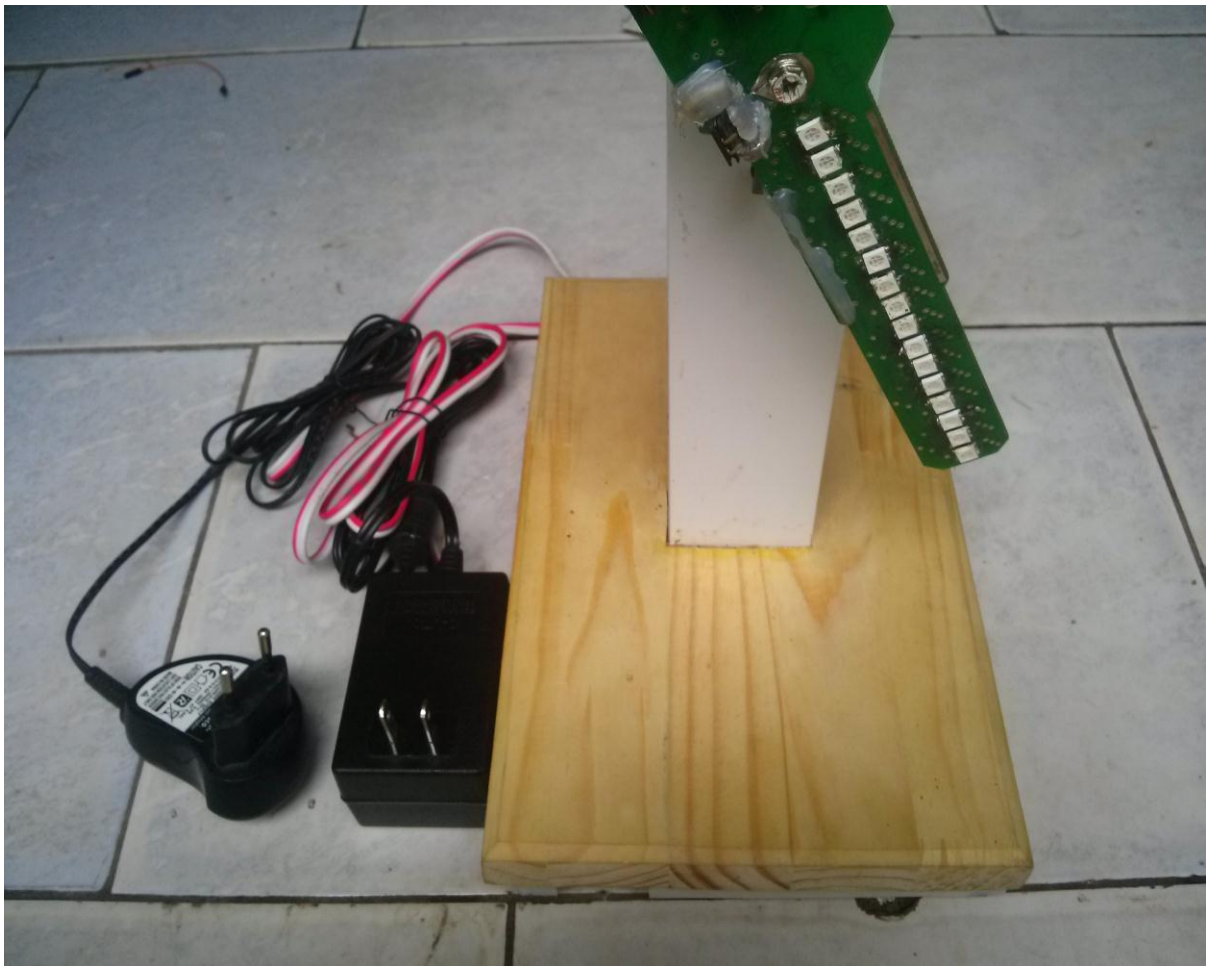


Figure 14: Whole propeller

1.3 Screen Design

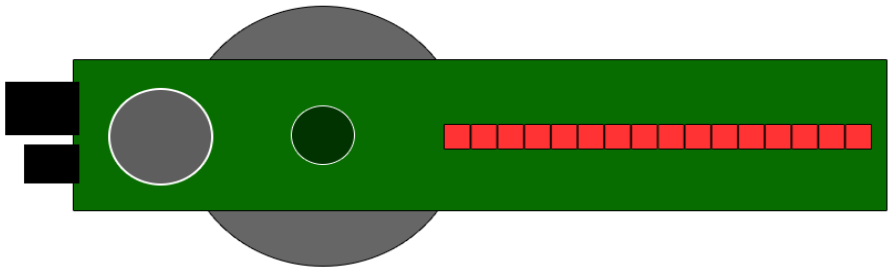


Figure 15: The board in stand-still position

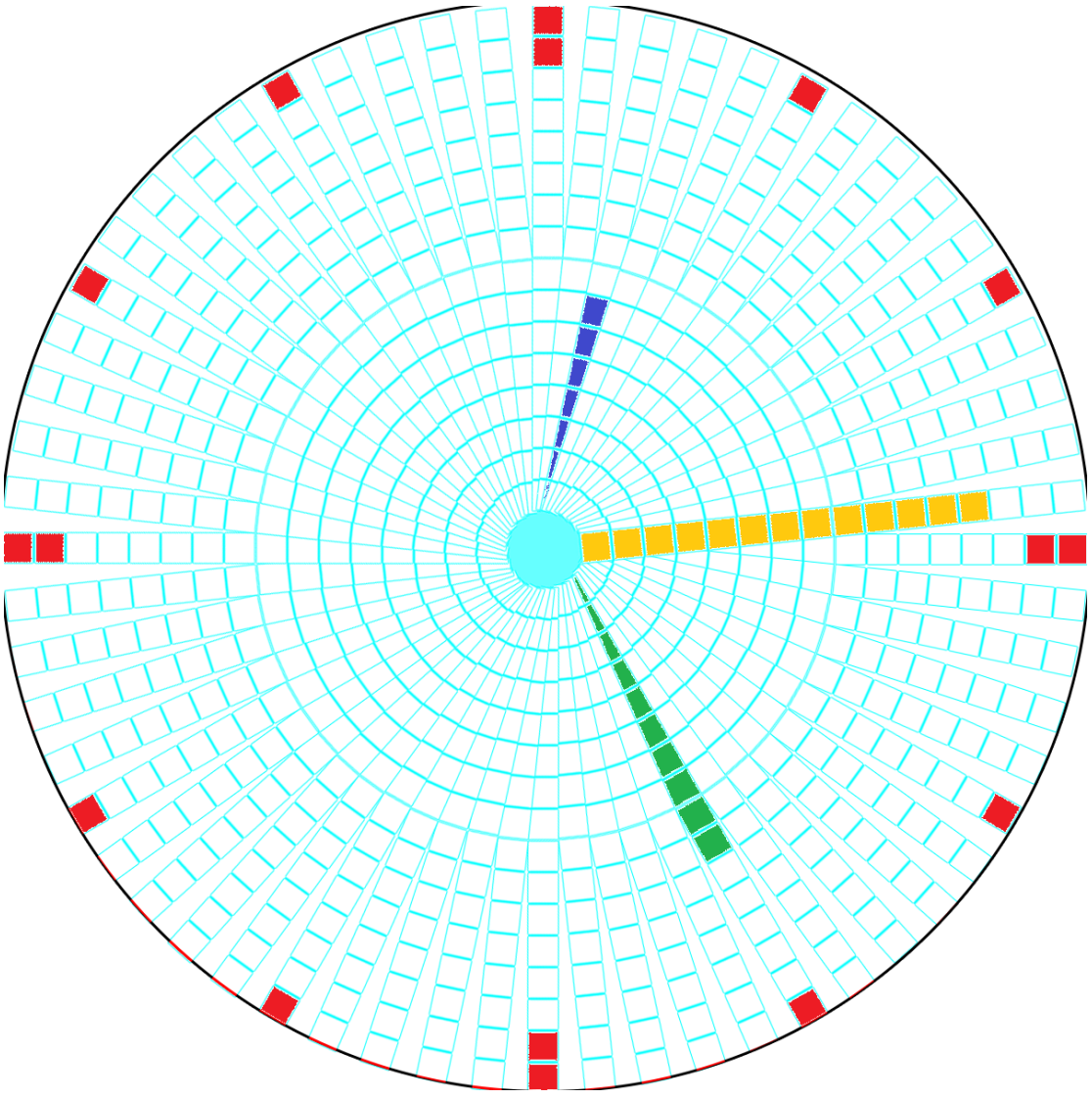


Figure 16: The board rotate and display an analog clock

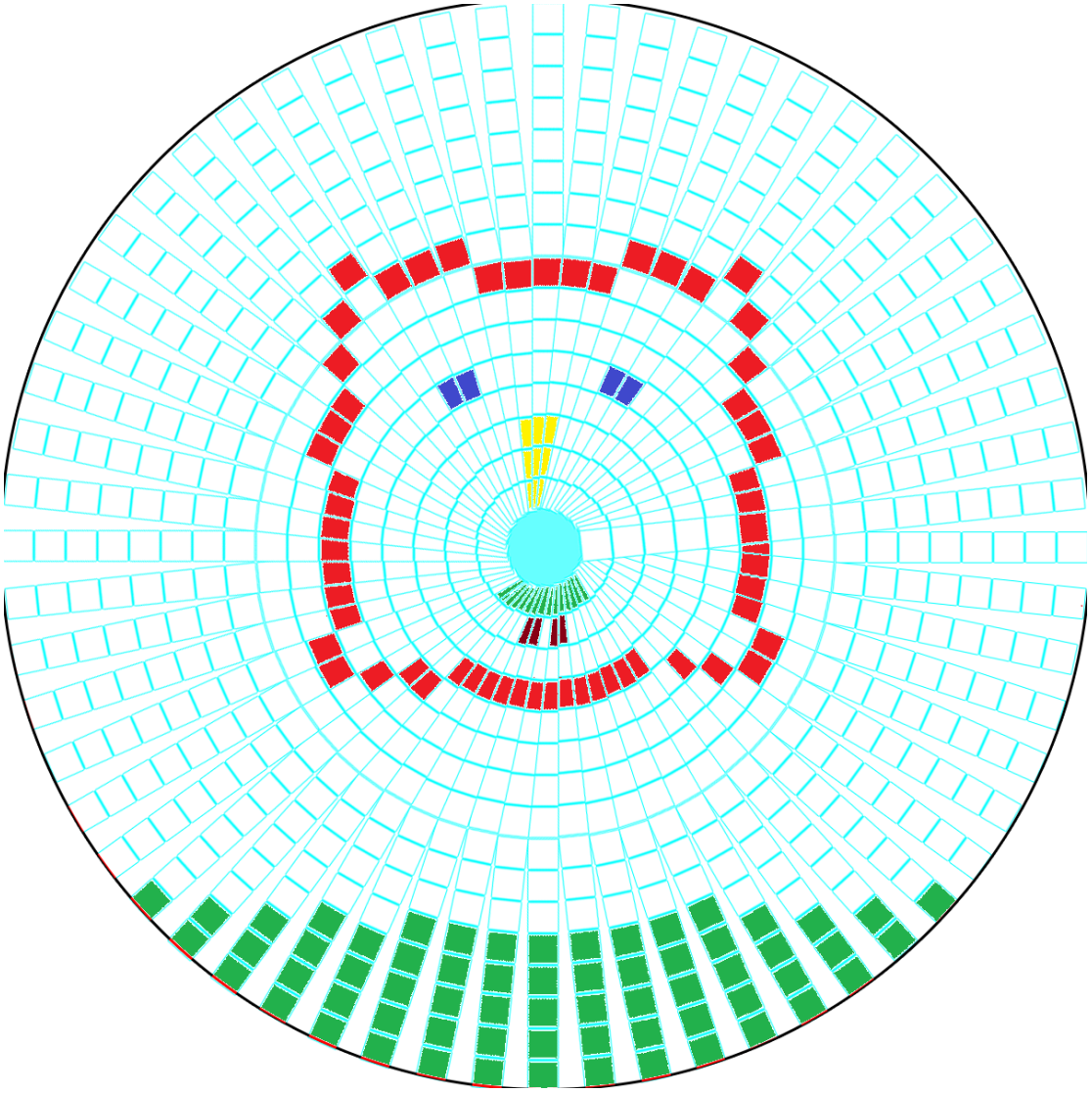


Figure 17: Display an image

2. Mechanical Design

The motor performance is very important in circuit design. The electrics motors directly affect its speed and pushing capability.

Repeated scanning of the display is necessary for continuous vision of image. This task is achieved using continuous circular rotation of the whole circuit assembly. Therefore a DC motor has used as the prime mover.

Figure 18: DC Motor

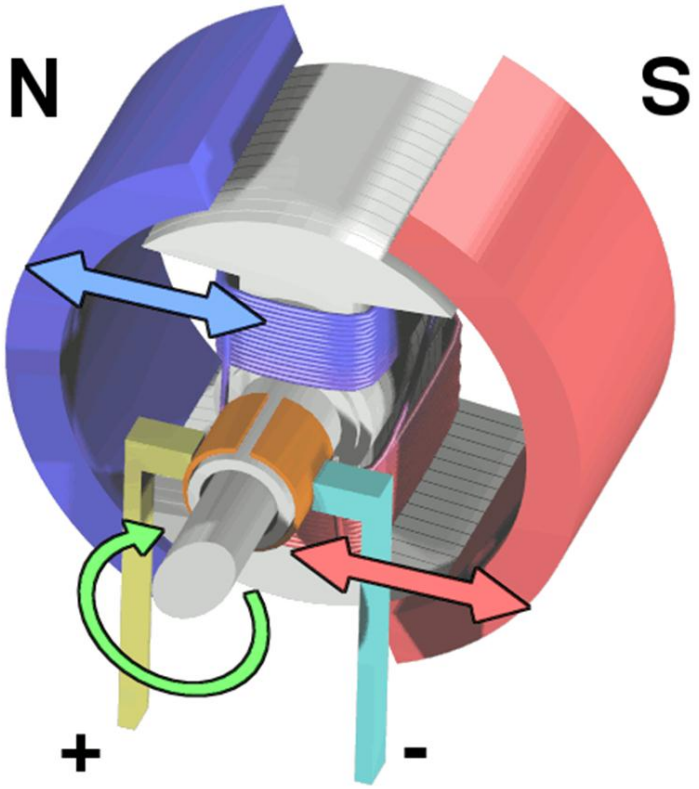


Figure 19: DC Motor Component

Workings of a brushed electric motor with a two-pole rotor (armature) and permanent magnet stator. "N" and "S" designate polarities on the inside faces of the magnets; the outside faces have opposite polarities. The + and - signs show where the DC current is applied to the commutator which supplies current to the armature coils.

DC motors looks quite simple. When applying a voltage to both terminals, it will spin. DC motors are non-polarized motors. Which means when changing the voltage direction, the motor will rotate in both directions, forward and backward. DC motors are rated from about 5V-12V. The larger motors are often 24V or more. But for the purpose of this project, it is necessary to use 5V-12V range motor. Voltage is directly related to torque of the motor. When increasing the supply voltage, the higher torque will be produce. And the more current supplied, the higher rpm (revolutions per minute) will be produce. Most of DC motors has low torque and high rpm. In this project, we use DC motor 5V-2A 1800rpm.

Design Motor Commutator:

Prepare: Motor, Rolling-element bearing, Nut & bolt, Soft plastics pipe, Bronze wire, Spring.

1. Motor

2. Rolling-element bearing

3. Nut and bolt

4. Spring

Figure 20: Rotation's components

Execute:

Hold rolling-element bearing into cylinder following this picture

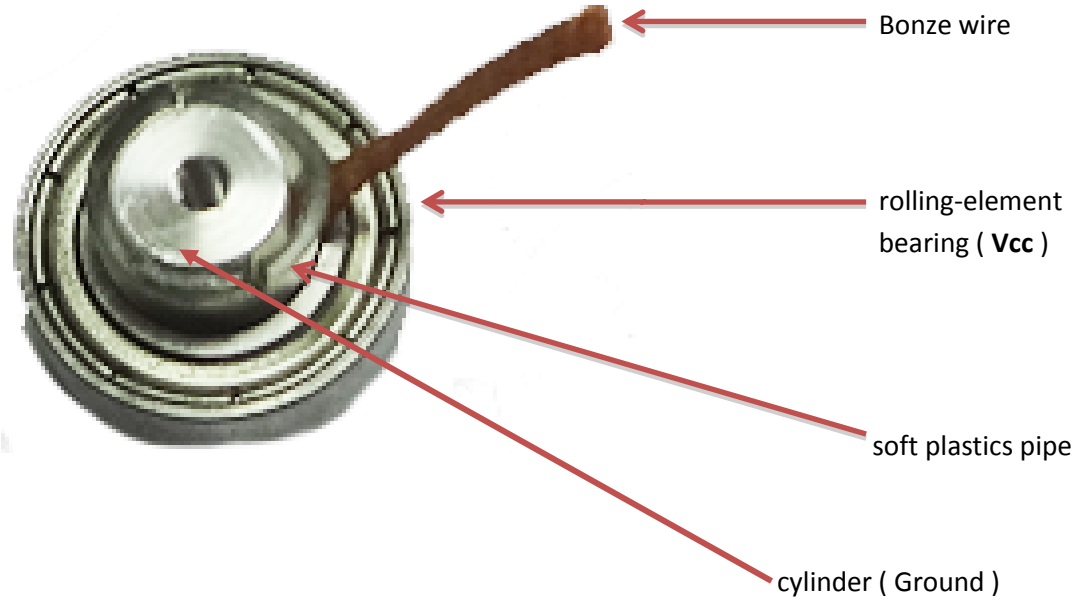


Figure 21: Assembly Rotation's components

Add spring around rolling-element bearing. We have product:

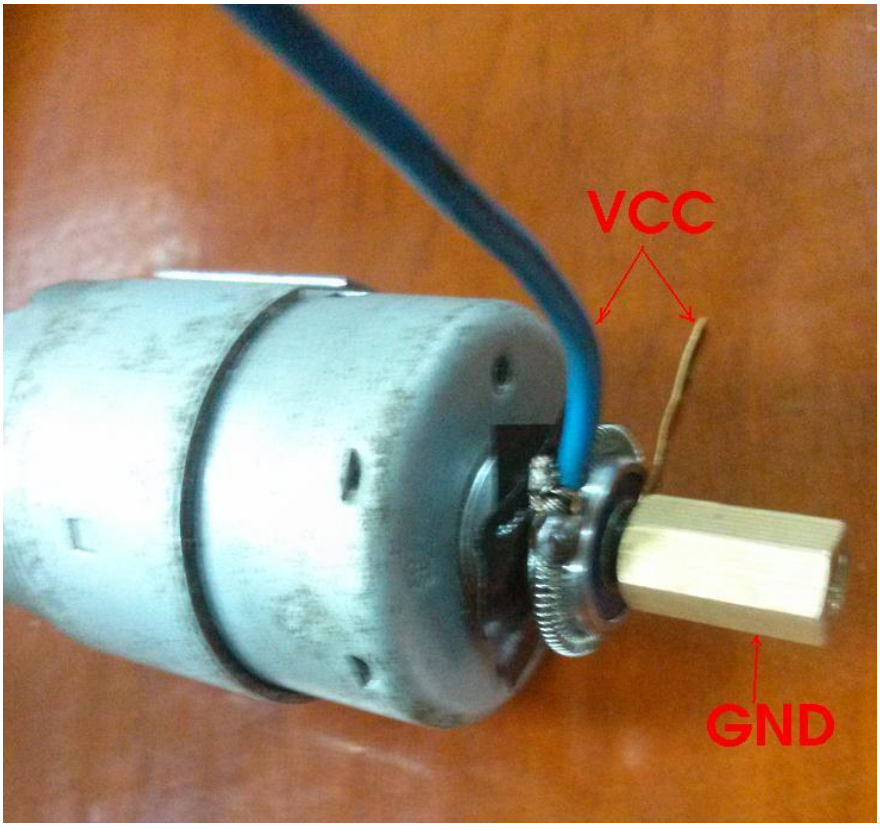
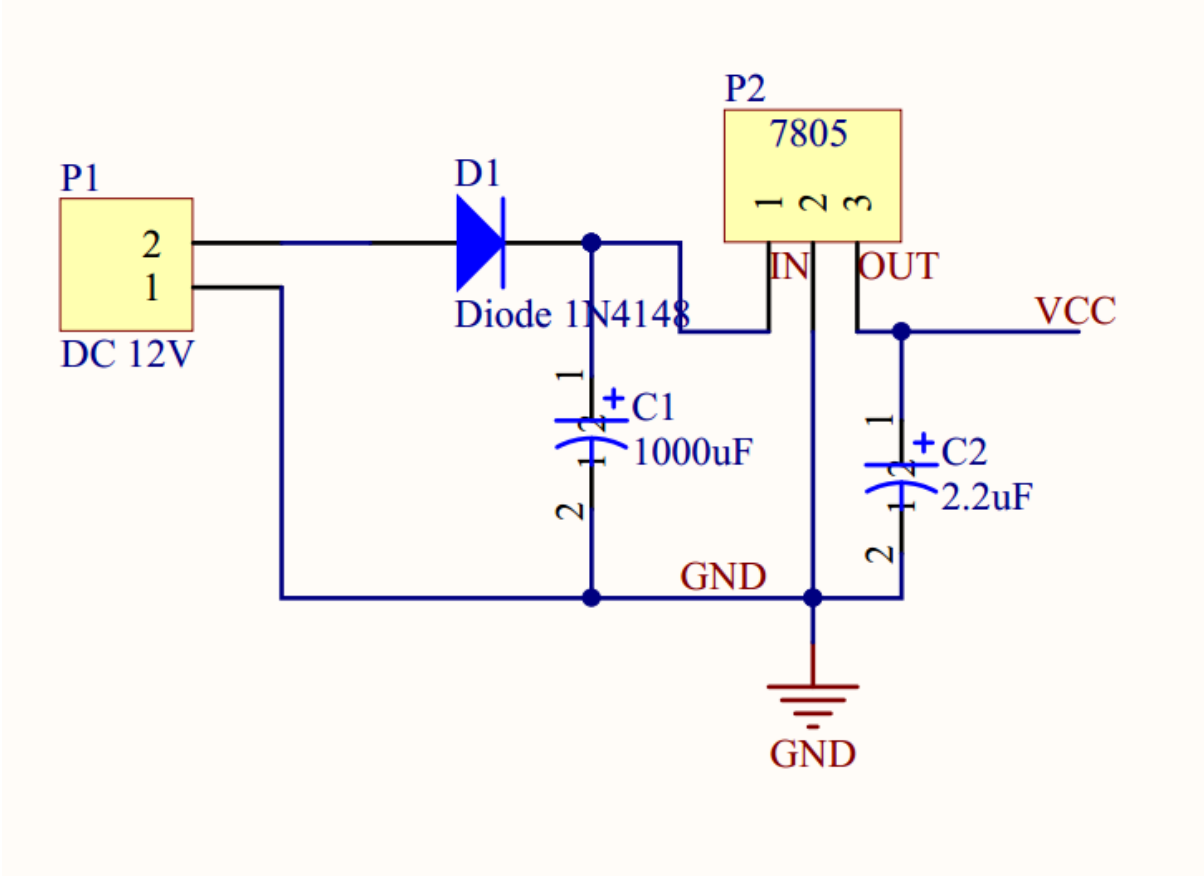


Figure 22: Completed Motor Commutator

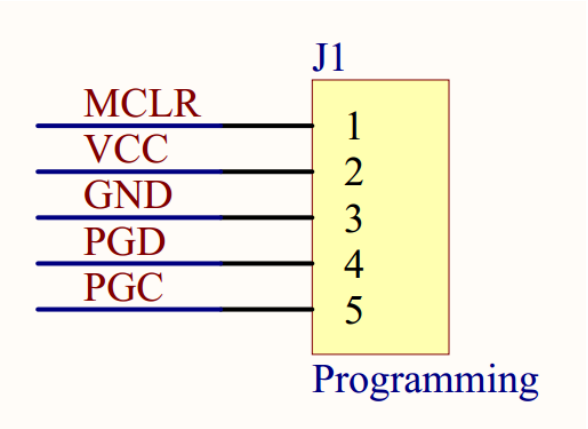
3. Propeller Design

1. Schematic

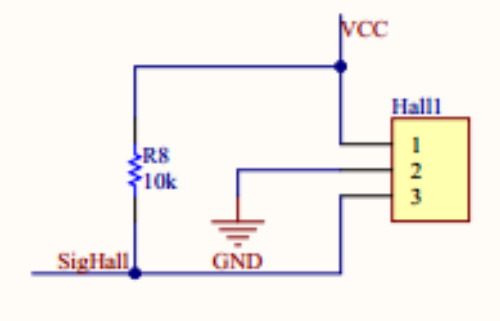
- Power Source 12V DC input and 5V DC output. Supply current to the whole propeller:



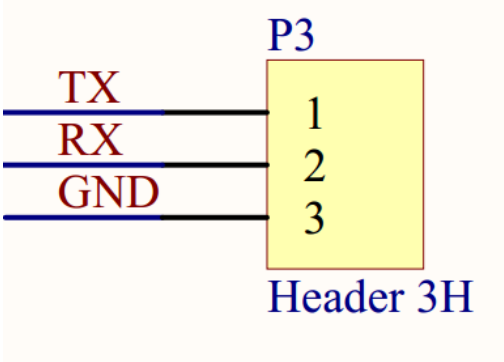
- ICSP - programming Micro Controller:



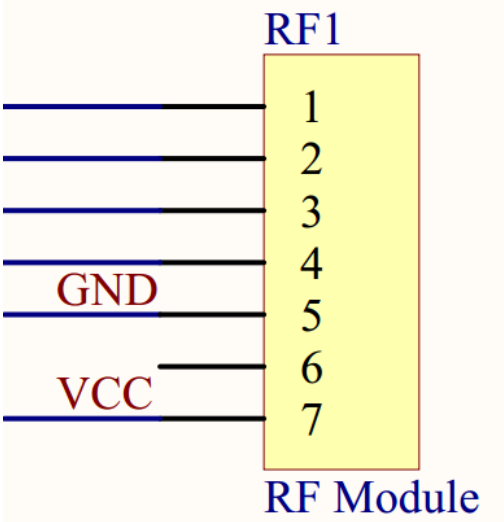
- Hall sensor - detect magnetic signal from magnet:



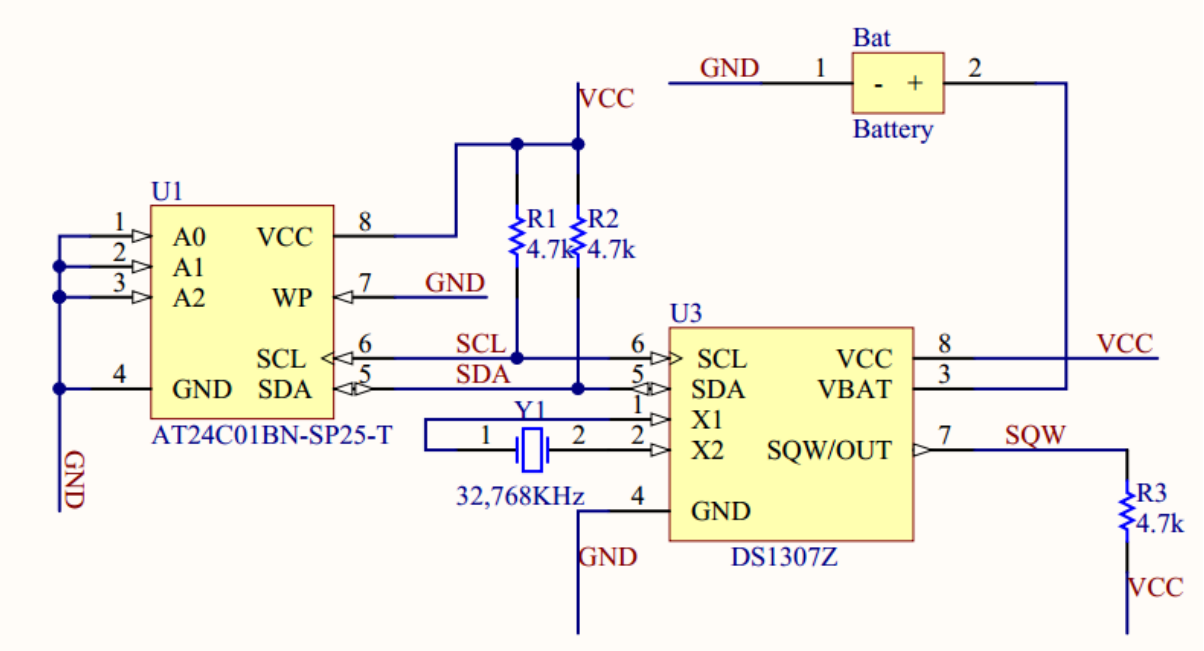
- RS232 - set time for Real-Time clock:



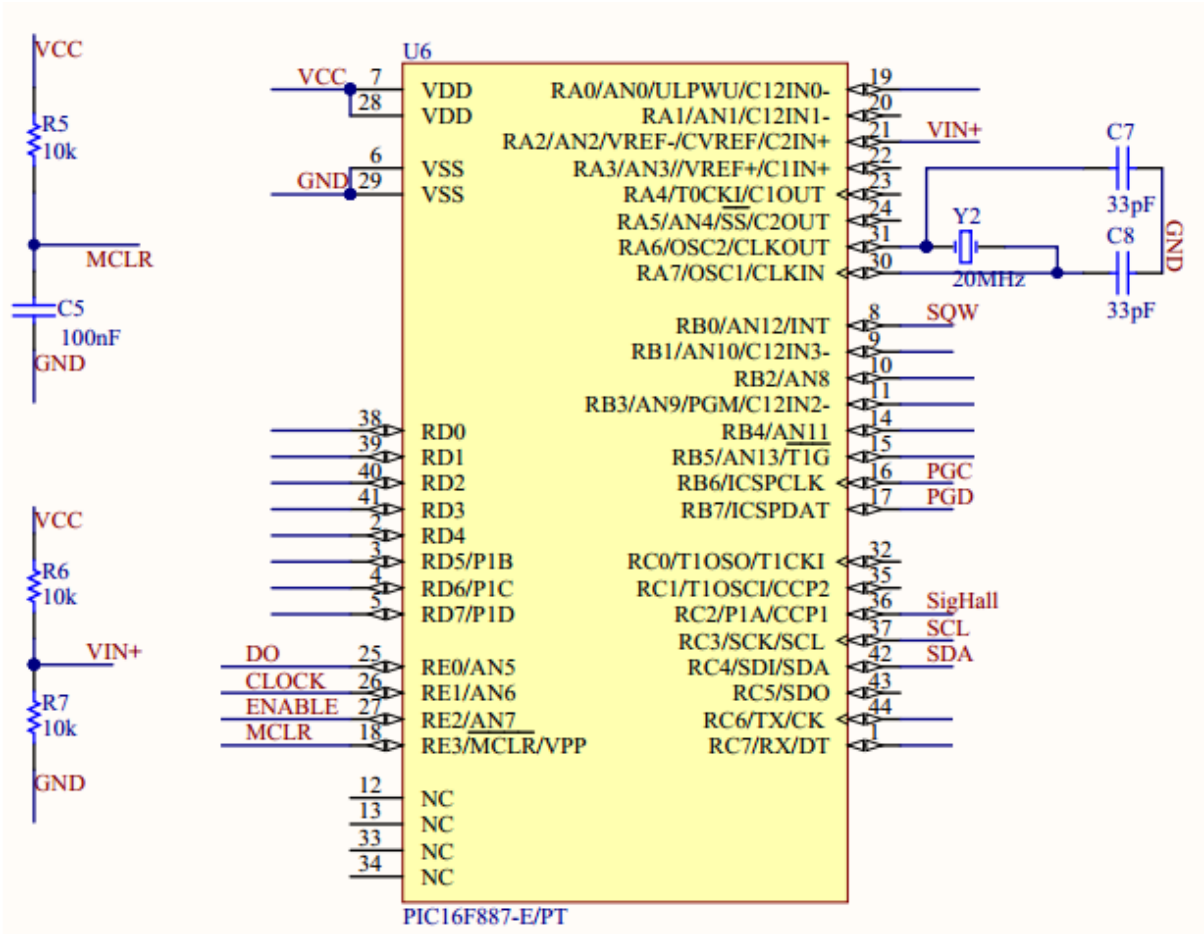
- RF receiver - receive signal from remote:



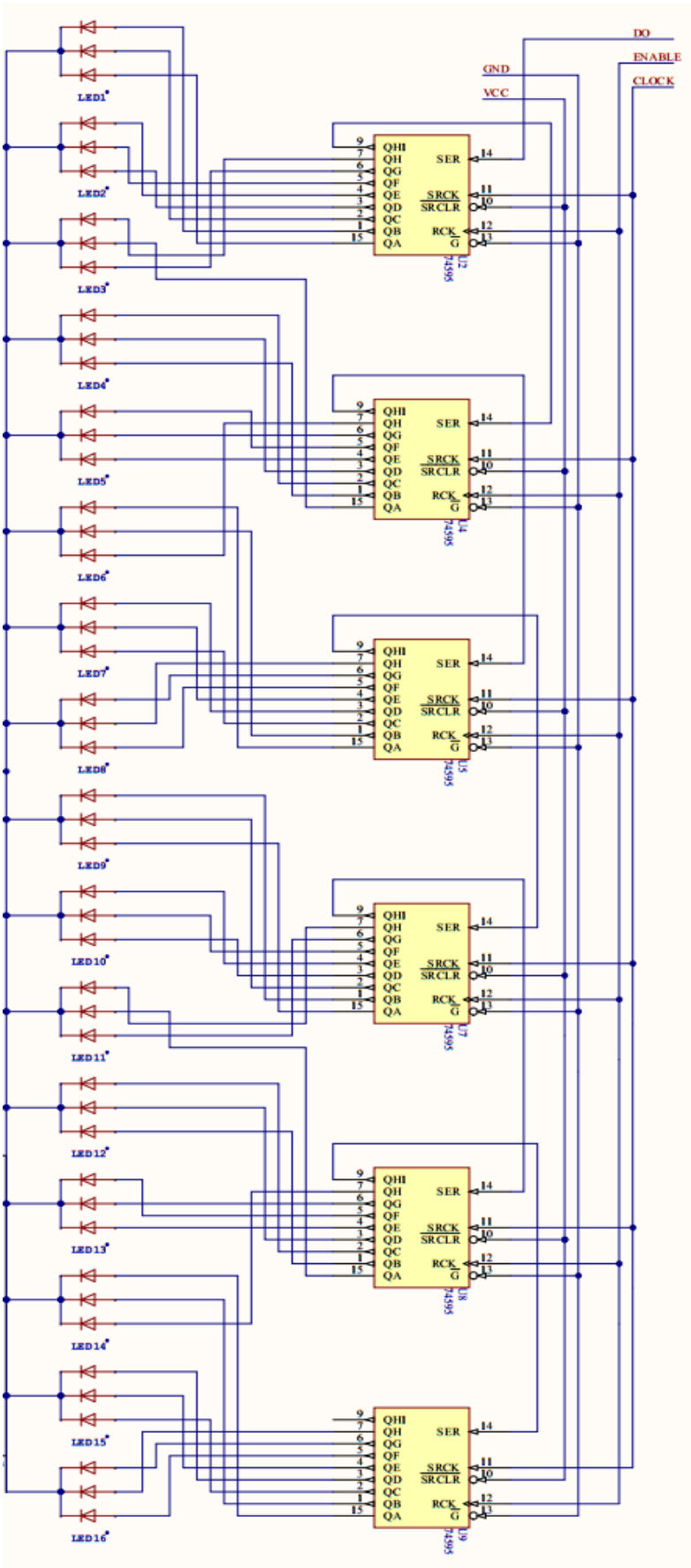
- EEPROM and Real-Time Clock:



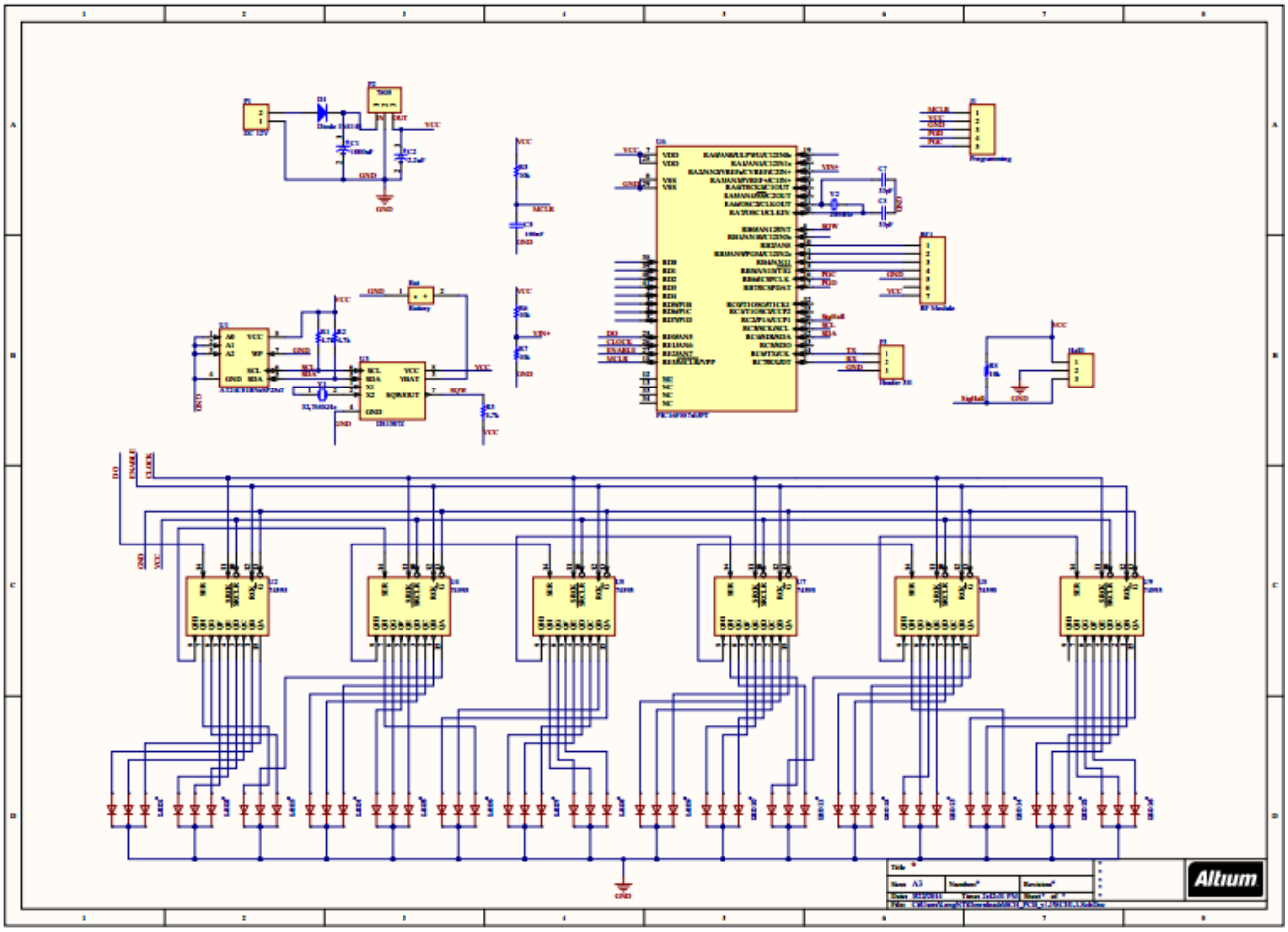
- Micro Controller:



- Shift Register and LED:



- The whole system:



2. Printed Circuit Board

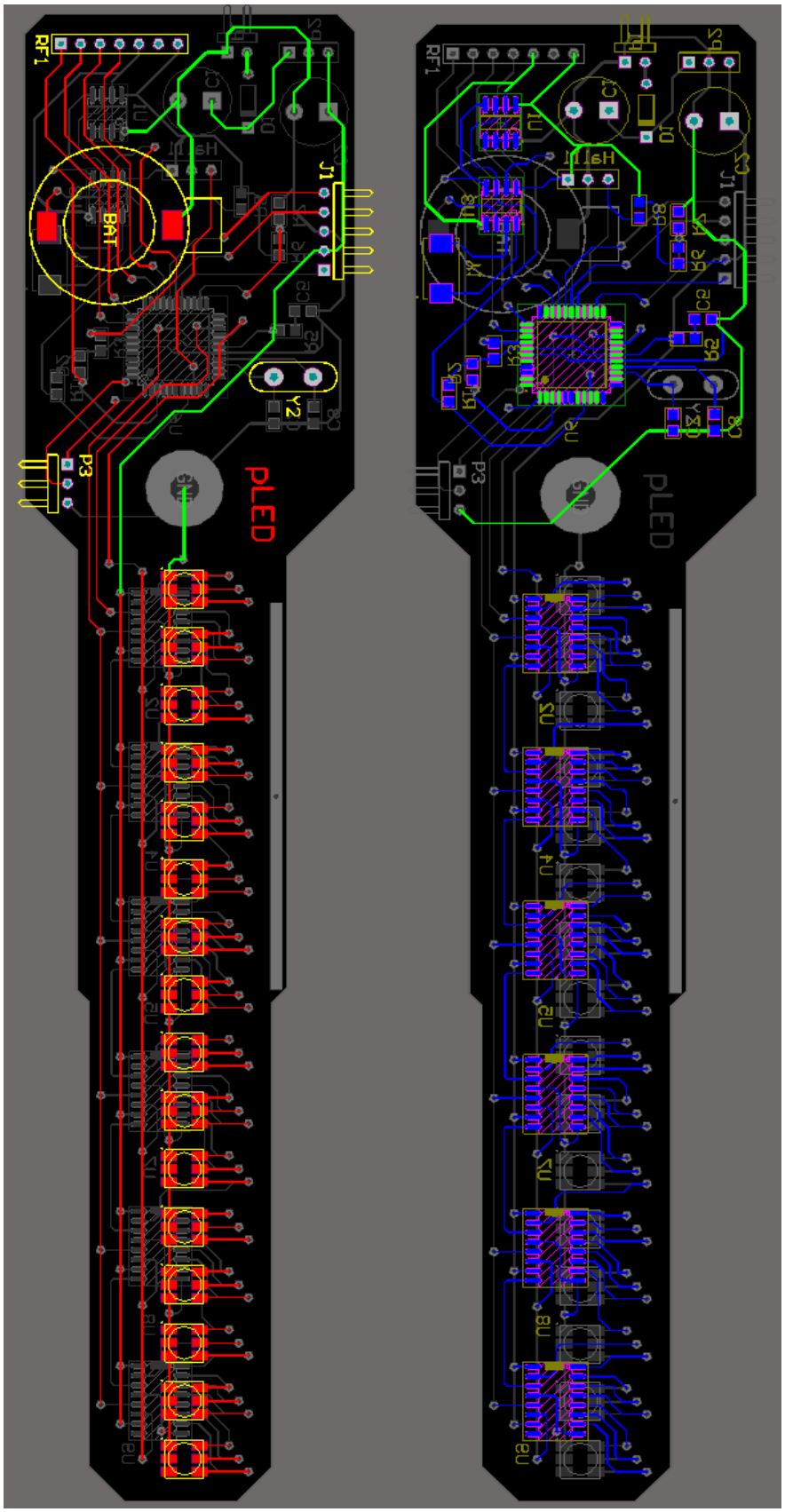


Figure 23: PCB Design

4. Firmware Design

1. Image Display

In one revolution, the board will rotate through 60 positions, called NODE. In each position, it will change the state of its LEDs. With 16 LEDs and 60 NODEs, we have a display resolution of 16 x 60.

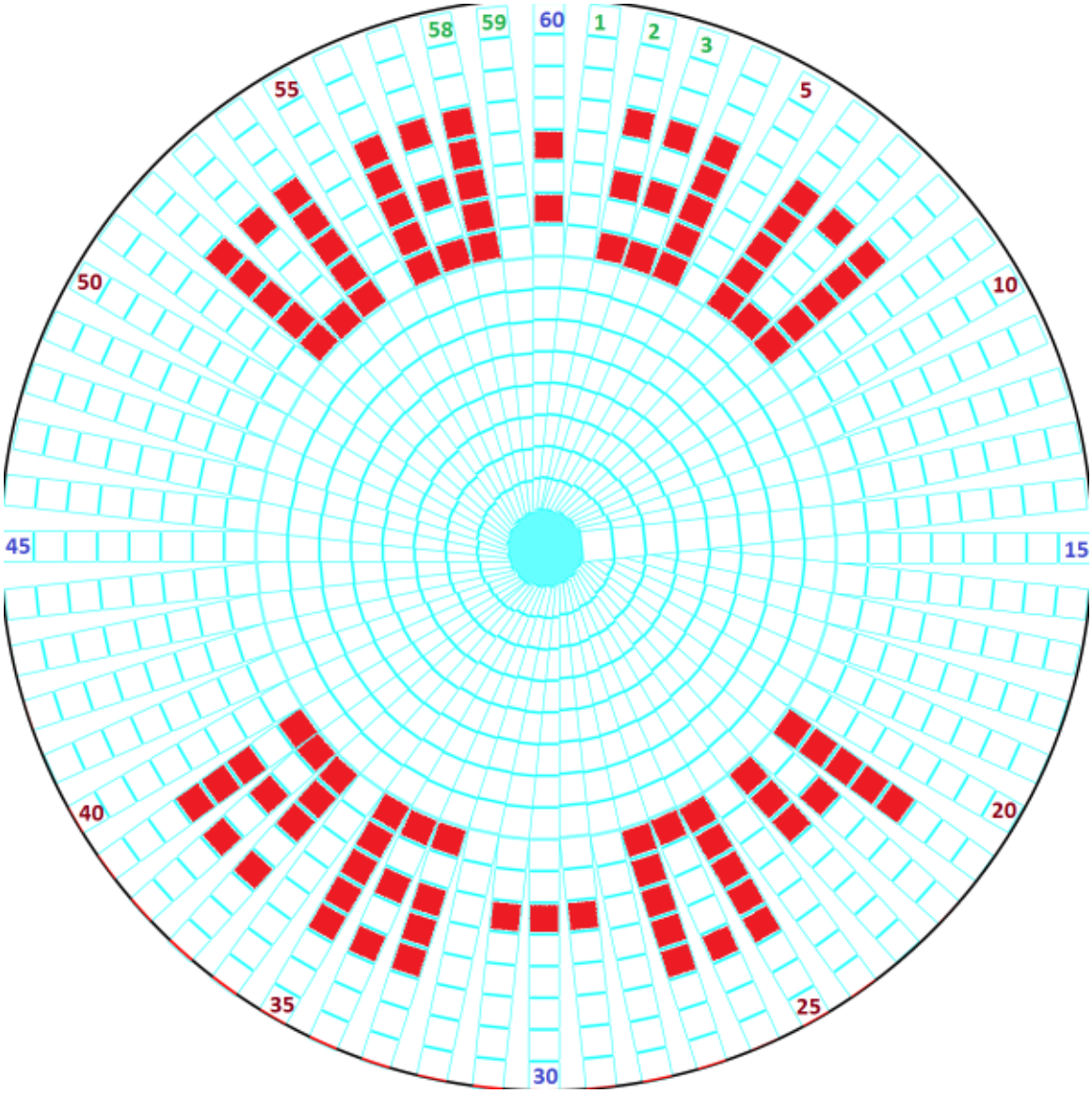


Figure 24: One display frame with resolution 16x60

To display one frame, the NODEs will be assigned a number correlative to its position in the frame. Base on this position number, the LEDs will be turned to an appropriate state of its color (red-green-blue). To calculate these states, we use two approaches:

1) With the analog clock, there are many NODEs which hold the same LEDs' state, so in each NODE, the LEDs are assigned a fixed state. Only with some special NODEs, as the hour, minute or second hand position, the LEDs are assigned with another state.

2) To display an image, the entire LEDs' state of 60 NODEs are stored in an array. In each NODE, based on its position number, the LEDs' state is recalled from the array.

2. Data Structure & Storage

A RGB LED has three independent states of its color: Red – Green – Blue. Then each LED must be represented by at least 3 bits of data. There are total 16 LEDs, to store entire state of these LEDs, it need 48 bits of data.

To accomplish the desired display resolution of 16 x 60 (16 LEDs at 60 different NODEs), we need to store 48 x 60 bits of data, which equivalent of 360 bytes. The PIC16F887 has only 368 bytes of RAM, many of which are used in various controlling purpose, the RAM is not suitable for storing image data. The display data should be store as constant variables in PIC's Program Memory. An external EEPROM chip is used to store extra data in case the PIC's memory is not enough.

A 2-byte variable is used to store one color state of all 16 LEDs. There are 3 variables for blue, red, green color. Those variables are stored consecutively in an array. So each NODE's state is stored in 3 elements of array. The array has 3 x 60 = 180 elements.

For the ease of calculation, the LEDs' current state is represented as a C's STRUCT

```
struct rgb_bits {int16 blue,  red,  green};
```

When the board rotates to a new NODE, rgb_bits' members are assigned new values from the array base on NODE's position.

For example: blue = 0x00FF; red = 0x0FF0; green = 0xFF00;

3. Display Algorithm

The main program can be divided into two parts: LEDs control and data manipulation.

3.1 LEDs Control

When system starts, the microcontroller (PIC16F887) will initiate variables, includes time values. Then the main program will go into an infinite while loop, where it uses 2 interrupts for displaying:

The first interrupt is CCP1, capture mode. The CCP1 pin receives input signal from Hall sensor. The period between two consecutive interrupt is stored in a variable which represent the period of propeller, the duration for the board to rotate exactly one revolution. That period will be divided by 60; the result is the duration of one NODE. A completed rotation will have 60 nodes of display LEDs.

The second one is Timer0 interrupt. This interrupt is used to trigger a display of one node and it is reset after each time interrupt occurs. The reset value is calculated from the period value in CCP1 interrupt.

LEDs are controlled by Shift Registers 74HC595. Before being shift to Shift Registers, data will be processed by a special function. This function will be triggered by interrupt signal from Timer0.

System uses RF module to change display mode. There are four output pins from RF module. They are connected to microcontroller through port B. When system receives signal from any output pins of RF module, the main program will change the value of "mode" variable, hence the display mode will be changed.

The time values (day, month, year, hour, minute, second) are stored as global variables and update each second through the use of external interrupt service routine. The RB0 pin (external interrupt) is connected with SQW/OUT pin of real time clock DS1307. The clock is configured to output a square wave pulse each second.

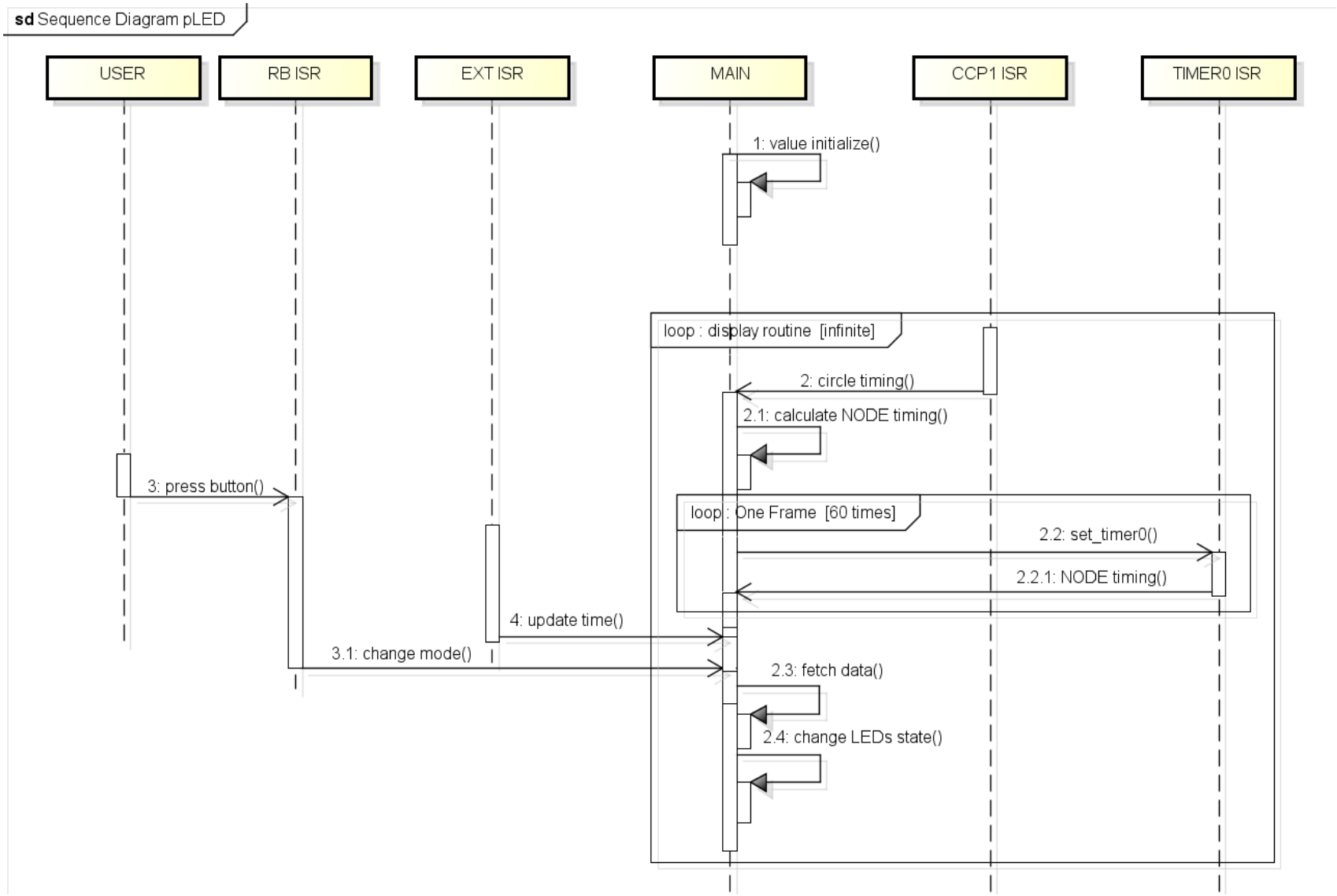


Figure 25: Sequence Diagram for Firmware

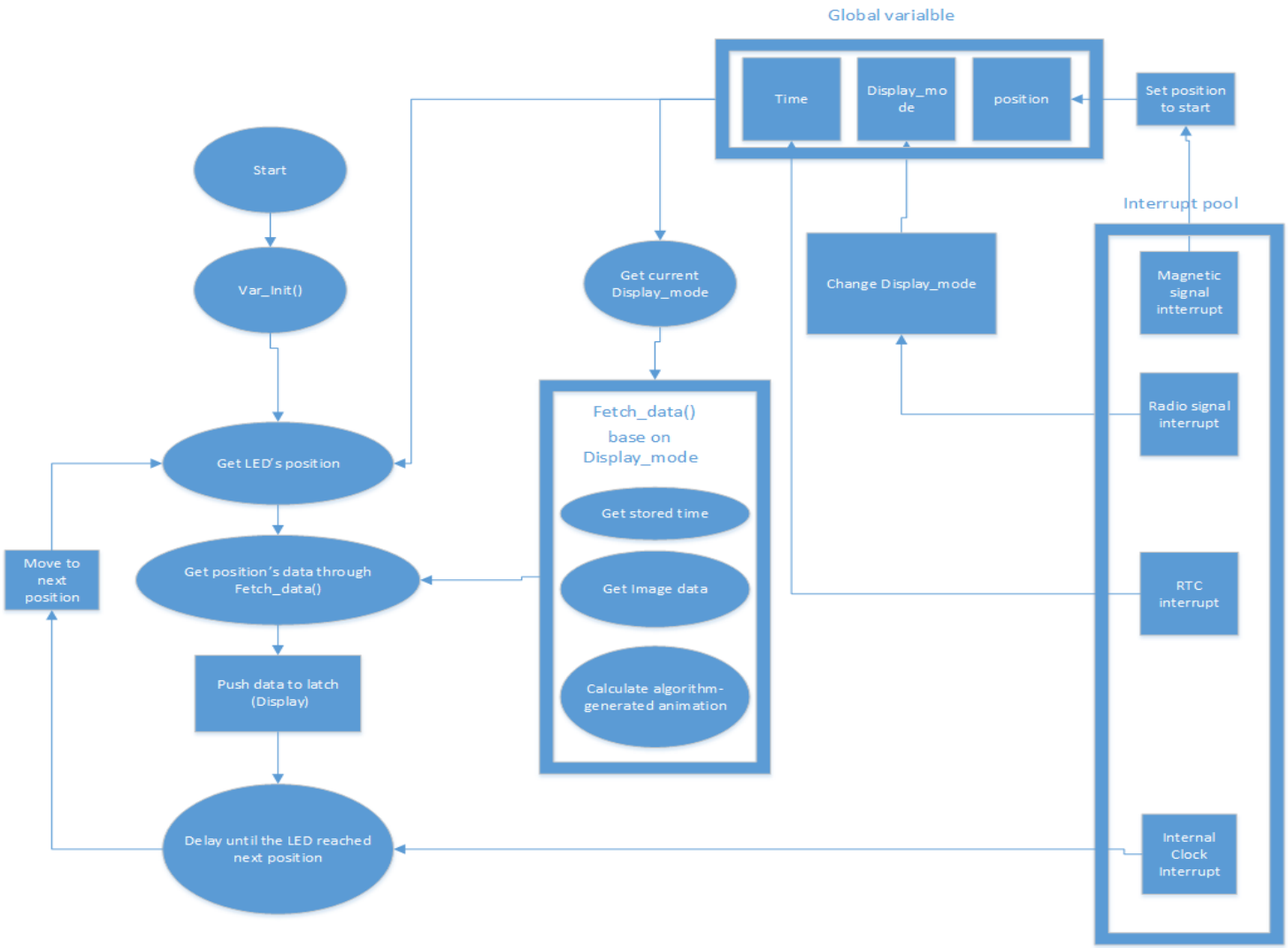


Figure 26: Firmware Flowchart

3.2 Data Manipulation

The fetch_data() function calculate data to send to Shift Registers. It uses several global variables to determine the data, including the NODE's position, display mode, date and time values.

A "mode" variable is used to determine display mode. In each mode, data is calculated in different way. There are 4 modes:

- MODE_A: data for analog clock. The data for analog clock is calculated directly. A fixed value is assigned for every NODE to display the clock's boundary. There are 12 NODEs that represent 12 hour values are assigned with different value. The hour, minute, second values are compared with NODE's position. If the result is a match, the rgb_bits is assigned with data that represent an hour, minute or second hand. Refer to Figure 16 – Screen Design for example.

- MODE_B: data for digital clock. Each number can be displayed with 3 x 5 LEDs, or data of 3 NODEs. It is only need to store data of one color. The 0 - 9 number characters is stored in a two dimensions array:

const int16 number[10][3];

Where number[x][y] represent an x number at position y. Date, time values can be displayed at fixed position on one frame. For example in Figure 24 – Image Display, the number 3 is display at NODE 2-3-4, the data can be retrieved at number[3][0], number[3][1], number[3][2].

- MODE_C: data for an image. Image data is stored in one dimension array:

const int16 image[180];

Data of consecutive NODEs is stored consecutively, with 3 bytes for one NODE. With variable NODE_position run from 1 to 60, the data for each NODE can be calculated:

rgb_bits.blue = image[NODE_position * 3 - 3];

rgb_bits.red = image[NODE_position * 3 - 2];

rgb_bits.green = image[NODE_position * 3 - 1];

Refer to Figure 17 – Screen Design for example.

- MODE_D: data for running letters. The letters can be stored as image data in MODE_C, but depend on the number of letters we can use array data with fewer elements to save memory space. For example, if each character needs 3 x 7 LEDs, the string "pLED" can be stored as:

const int16 pled [15];

With a variable pled_position, we can change the display position of “pLED” instead of fix position as in MODE_B. After a number of frames, pled_position is changed by a small amount in range of 1 – 60.The string “pLED” will appear as it’s running in circle.

CHAPTER 5: IMPLEMENTATION & TESTING

1. Implementation

1.1 Code structure and function explanation

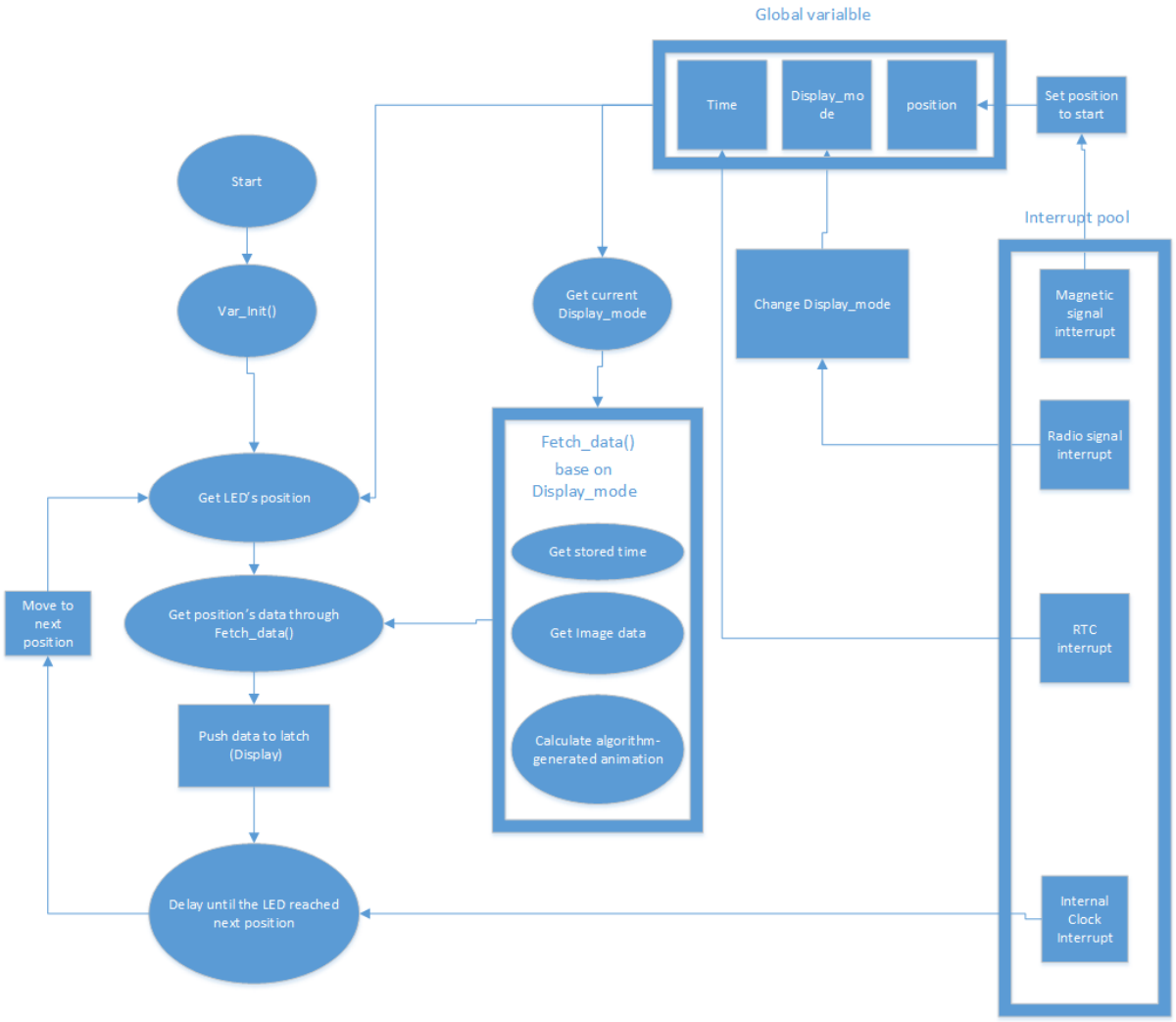


Figure 27: System structure diagram

1.1.1 Fetch_data( ) for clock

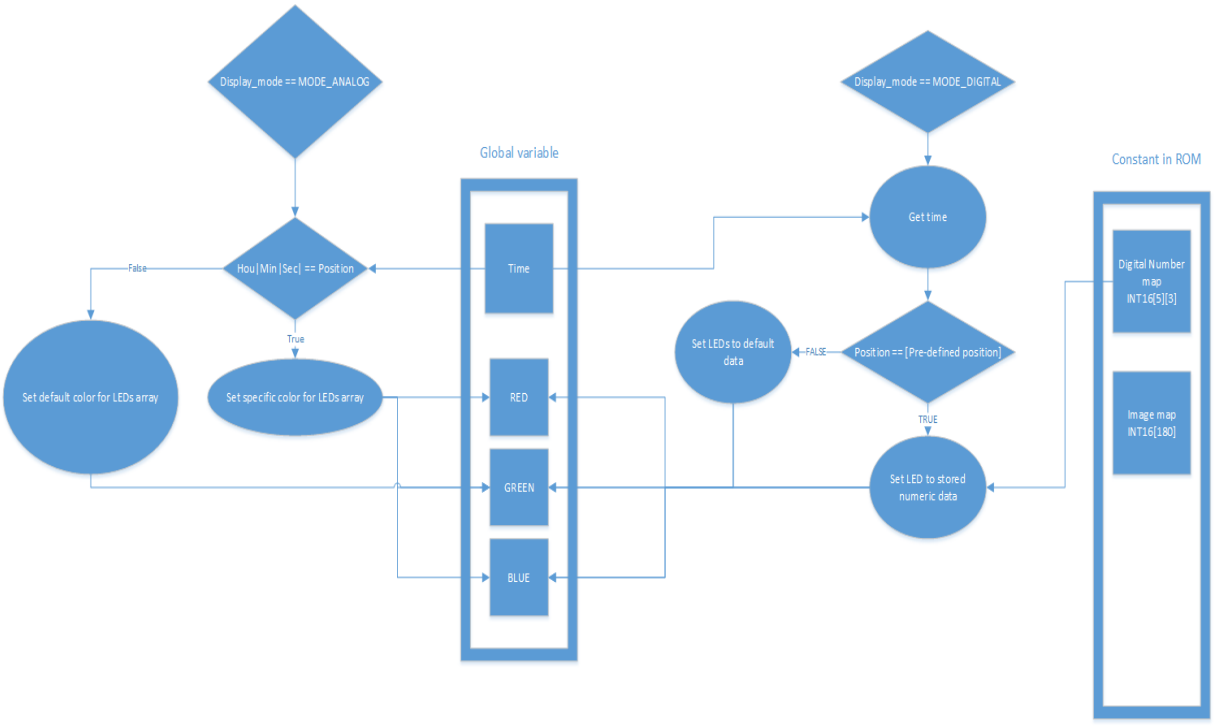


Figure 28: Get data for Analog and Digital

```
//   There are four modes:                              ||
//                                - MODE_A: data for analog clock          ||
//                                - MODE_B: data for digital clock         ||
//                                - MODE_C: data for an image, FPT logo    ||
//                                - MODE_D: data for running letters, pLED ||
//=========================================================||
void fetch_data()
{
        If (mode == MODE_A) // display analog clock
        {
        // The section_count vary from 1-60 and need to adjust to 0-59 value of
minute    and second
                int8 pos = section_count;
                if (section_count == 60)       pos = 0;
                // set default value for normal section
                rgb_bits.blue =  0x0001;
                rgb_bits.red =   0x0001;
                rgb_bits.green = 0x0001;
                // leds' value for hour mark on analog clock
                if((section_count % 5) == 0){
                        rgb_bits.blue =  0x0003;
                        rgb_bits.red =   0x0003;
                        rgb_bits.green = 0x0003;
                }
                // leds' value at 3-6-9-12 hour
                if((section_count % 15) == 0){
                        rgb_bits.blue =  0x007;
                        rgb_bits.red =   0x007;
                }
                // hour hand's position
                if(pos == anal_hour){
                        rgb_bits.blue = 0xFC00;
                }
                // minute hand's position
                if(pos == min){
                        rgb_bits.red = 0xFF00;
                }
                //second hand's position
                if(pos == sec){
                        rgb_bits.green = 0xFFF1;
                }
        }
```

1.1.2 Fetch_data() for Image

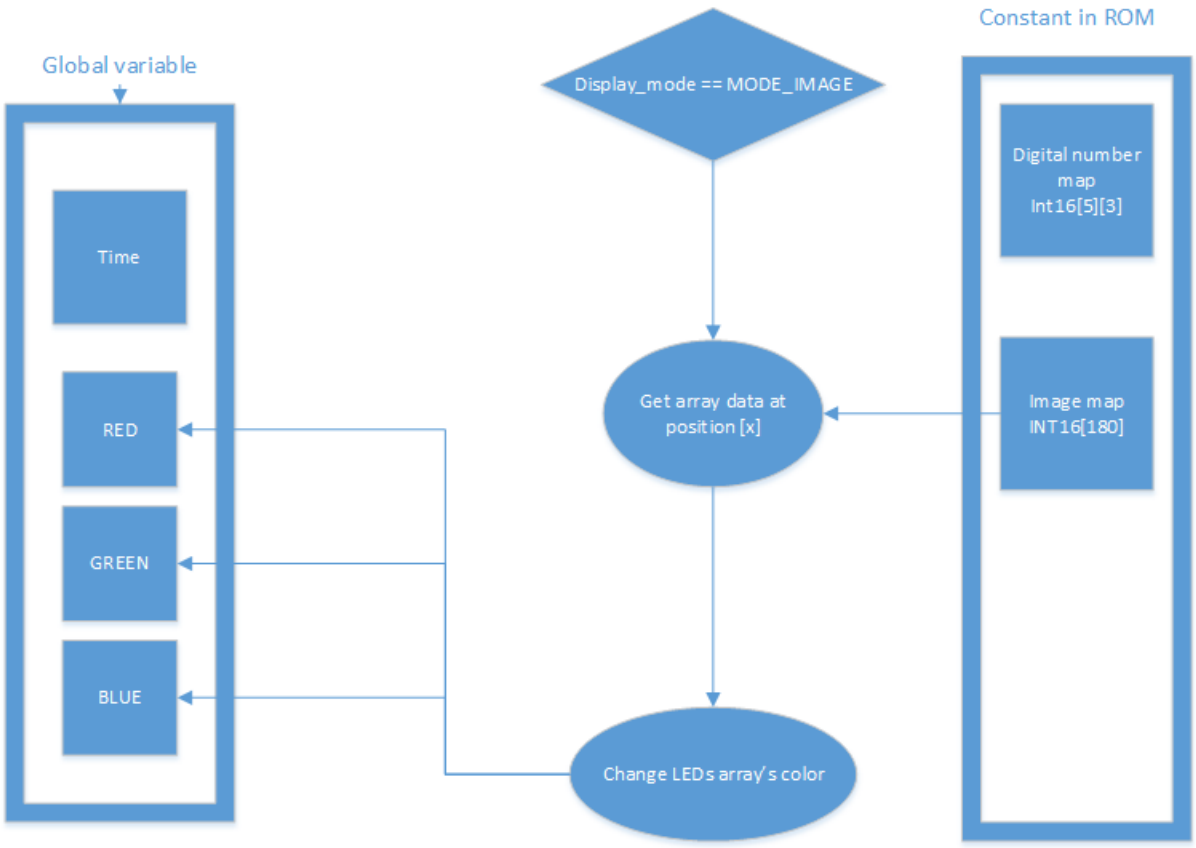


Figure 29: Get data for Image display

```
If (Display_mode == MODE_C)   // display an image - FPT logo
        {
                //fpt[180] is the image data, which store information for entire circle
                rgb_bits.blue = fpt[section_count * 3 - 3];
                rgb_bits.red = fpt[section_count * 3 - 2];
                rgb_bits.green = fpt[section_count * 3 - 1];
        }
```

1.1.3 Rotation timing

The system use 2 interrupts to acknowledge when a new rotation started and when to print out the next Position's data of LEDs array.

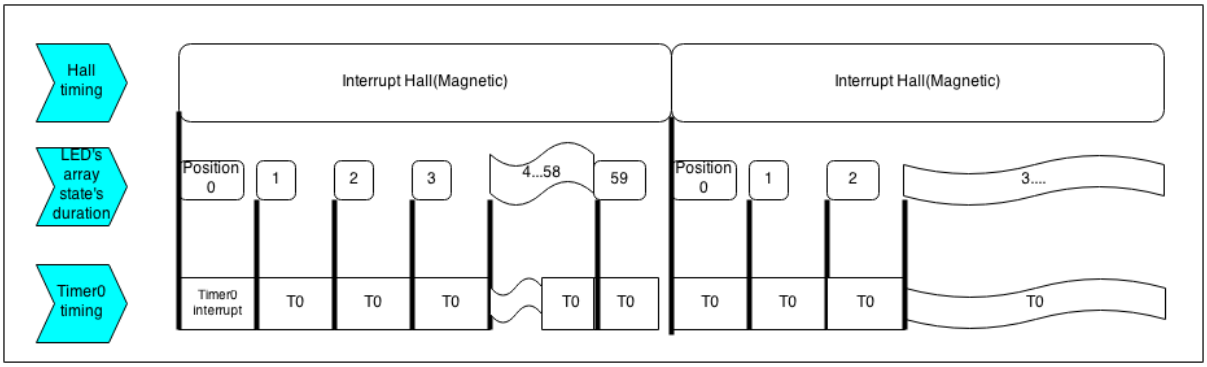


Figure 30: Duration estimate for Timer0

Hall and Timer1 interrupt are used together to calculate precise timing for Timer0.

```
void  TIMER1_isr(void) // run on timer1 overflow
{
  timer1_overflow++;  //  number of times overflow counter
}
```

[sigHall_timer] represent *duration of an* LEDs array rotation.

```
void  CCP1_isr(void)  //  run on HALL interrupt
{
  sigHall_timer = ((int32)timer1_overflow * 0xFFFF + CCP_1);//  overflow times multiply with
number      of circle(65536), plus number of circle left in timer1. Will convert to real time unit
later
  timer1_overflow = 0; // reset number of overflow counter
  set_timer1(0);   //reset timer1
}
```

1.2 Driver Optimization

The firmware is developed using CCS Compiler which provides most of driver code for 74595 IC, 24256 memory chip. But during implementation, we realize the PIC16F887 calculation speed is not enough to display 16 RGB LEDs at the motor speed.

To determine the rotation speed, we used a separate code for firmware:

```
setup_timer_1(T1_INTERNAL | T1_DIV_BY_2);      //div 2 - 26.2 ms overflow
setup_ccp1(CCP_CAPTURE_FE);
int i;
int8  count = 0;
int8* pt_byte;
while(true)
{
        if(count == 60)
        {
                write_eeprom(count, 0xFF);
                write_eeprom(count + 1, bin2bcd(count));
                count = 0;
        }
                if(cycle_trigger == 1)
                {
                  pt_byte = &sigHall_timer;
                  //write int32 sigHall_timer to eeprom using int8 pointer
                  write_eeprom(count, *(pt_byte + 3));
                  write_eeprom(count + 1, *(pt_byte + 2));
                  write_eeprom(count + 2, *(pt_byte + 1));
                  write_eeprom(count + 3, *(pt_byte + 0));
                  count += 4;
                 cycle_trigger = 0;
                }
}
```

The sigHall_timer value is stored in the PIC's internal EEPROM 15 times. The sigHall_timer is the number of time the timer1 change its value. Timer1 is configured using internal clock Fosc = 20MHz. In this mode, timer1 uses ¼ of system clock. With prescaler divided by 2, the frequency which timer1 changes its value is:

$$Fosc / 8 = 2.5MHz$$

The duration for one rotation:

$$Circle\_Time = sigHall\_timer / 2.5\ (us)$$

Speed of the motor:

$$Motor\_Speed = 10^6 / Circle\_Time\ (Round\ Per\ Second)$$

Time of test: 11h30' 3-April		
Adapter 5V - 2A		
sigHall_timer	Circle_Time(us)	Motor_Speed(RPS)
81555	32622	30.65415977
81563	32625.2	30.6511531
81574	32629.6	30.64701988
81592	32636.8	30.64025885
81590	32636	30.64100993
81769	32707.6	30.57393389
81516	32606.4	30.66882575
81495	32598	30.67672863
81468	32587.2	30.68689547
81453	32581.2	30.69254662

Average duration of one circle: Circle_Time ≈ 32600 us, one NODE is displayed in:

NODE_Time = Circle_Time / 60 ≈ 543 us

With Fosc = 20MHz, PIC16F887 instruction cycle is 5 MIPS (5 x 106 instructions per second) or 5 instructions per microsecond. In duration of NODE_Time, the PIC can perform about 5 x 543 ≈ 2500 instruction cycles.

The CCS's 74595.h driver using function

void write_expanded_outputs(BYTE eo);*

to perform shifting operation. This function can be used for shifting any number of Shift Registers with the BYTE pointer. Using CCS's listing function we calculated with 6 Shift Registers, this function needs about 2800 instructions cycles. With the addition of fetch_data() function and the calculating of controlling variables, the PIC16F887 microcontroller cannot execute all instruction cycles in the NODE_Time duration.

The optimized 74595.h driver use function:

void latch_write(int16 blue, int16 red, int16 green);

which removed the use of pointer and fix the number of Shift Registers to 6 as in schematic design. The result is the new shift function only use about 1400 instruction cycles, enough for the PIC to execute within NODE_Time duration.

The latch_write() function also change the parameters and shifting order that is suitable with rgb_bits data structure.

Optimized driver

```
void latch_write(int16 blue, int16 red, int16 green){
        BYTE i;
        output_low(EXP_OUT_CLOCK);
        output_low(EXP_OUT_ENABLE);
        for(i = 16; i > 0; i--)     {
  //write green led first, the first bit output will be on the last latch
                if(green & 0x01)// green
                   output_high(EXP_OUT_DO);
                else
                  output_low(EXP_OUT_DO);
                shift_right(&green, 2, 0);
                output_high(EXP_OUT_CLOCK);
                output_low(EXP_OUT_CLOCK);

                //output red
                if(red & 0x01)
                        output_high(EXP_OUT_DO);
                else
                        output_low(EXP_OUT_DO);
                shift_right(&red, 2, 0);
                output_high(EXP_OUT_CLOCK);
                output_low(EXP_OUT_CLOCK);

                //output blue
```

Pre-Optimized Drive

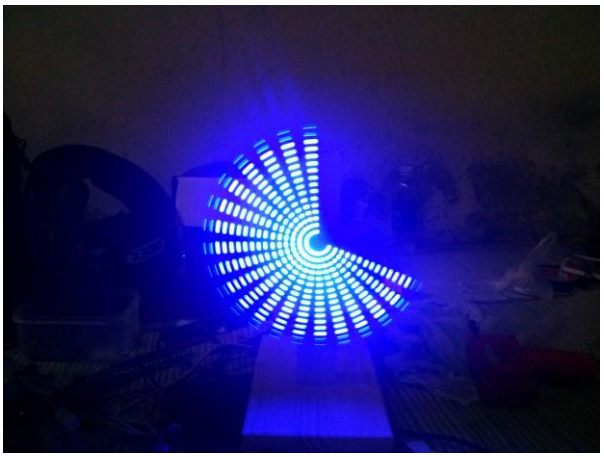
```
void write_expanded_outputs(BYTE* eo) {
 BYTE i;

 output_low(EXP_OUT_CLOCK);
 output_low(EXP_OUT_ENABLE);

 for(i=1;i<=NUMBER_OF_74595*8;++i) {  // Clock out bits from the eo
array
   if((*(eo+(NUMBER_OF_74595-1))&0x80)==0)
    output_low(EXP_OUT_DO);
   else
    output_high(EXP_OUT_DO);
  shift_left(eo,NUMBER_OF_74595,0);
  output_high(EXP_OUT_CLOCK);
  output_low(EXP_OUT_CLOCK);
 }
 output_high(EXP_OUT_ENABLE);
 output_low(EXP_OUT_ENABLE);
}
```

```
                    if(blue & 0x01)
                        output_high(EXP_OUT_DO);
                    else
                        output_low(EXP_OUT_DO);
                    shift_right(&blue, 2, 0);
                    output_high(EXP_OUT_CLOCK);
                    output_low(EXP_OUT_CLOCK);
                }
                output_high(EXP_OUT_ENABLE);
                output_low(EXP_OUT_ENABLE);
}
```

Change LED's state 40 times in 2/3 rotation



Change LED's state 40 times in 1/4 rotation

1.3 Real Time Clock Configuration

DS1307.h driver is based on CCS forum's suggestion. Refer to reference [4].

For usage in this project, we rewrite some configuration for DS1307 clock. In DS1307_init() function, the CH bit is clear to ensure the clock running using oscillator. SQW/OUT pin is enabled with frequency 1Hz. In ds1307_set_date_time() function, the hour byte is configured to use 24h format.

1.4 Configure tool

PC application is used to configure date, time of real time clock on the board. It is also used for creation of image data:

- Set time: by the use of RS232 communication, date, time data can be sent to the board and the PIC microcontroller can use this data to configure the DS1307 real time clock.

- Create image data: draw an image on the panel and the software can generate array data of the image. Currently, this image data must be update in the firmware code and recompile to change display image of Propeller LED. If the

product needs to be extended its feature, this data can be sent to the board using serial communication as the date, time data.

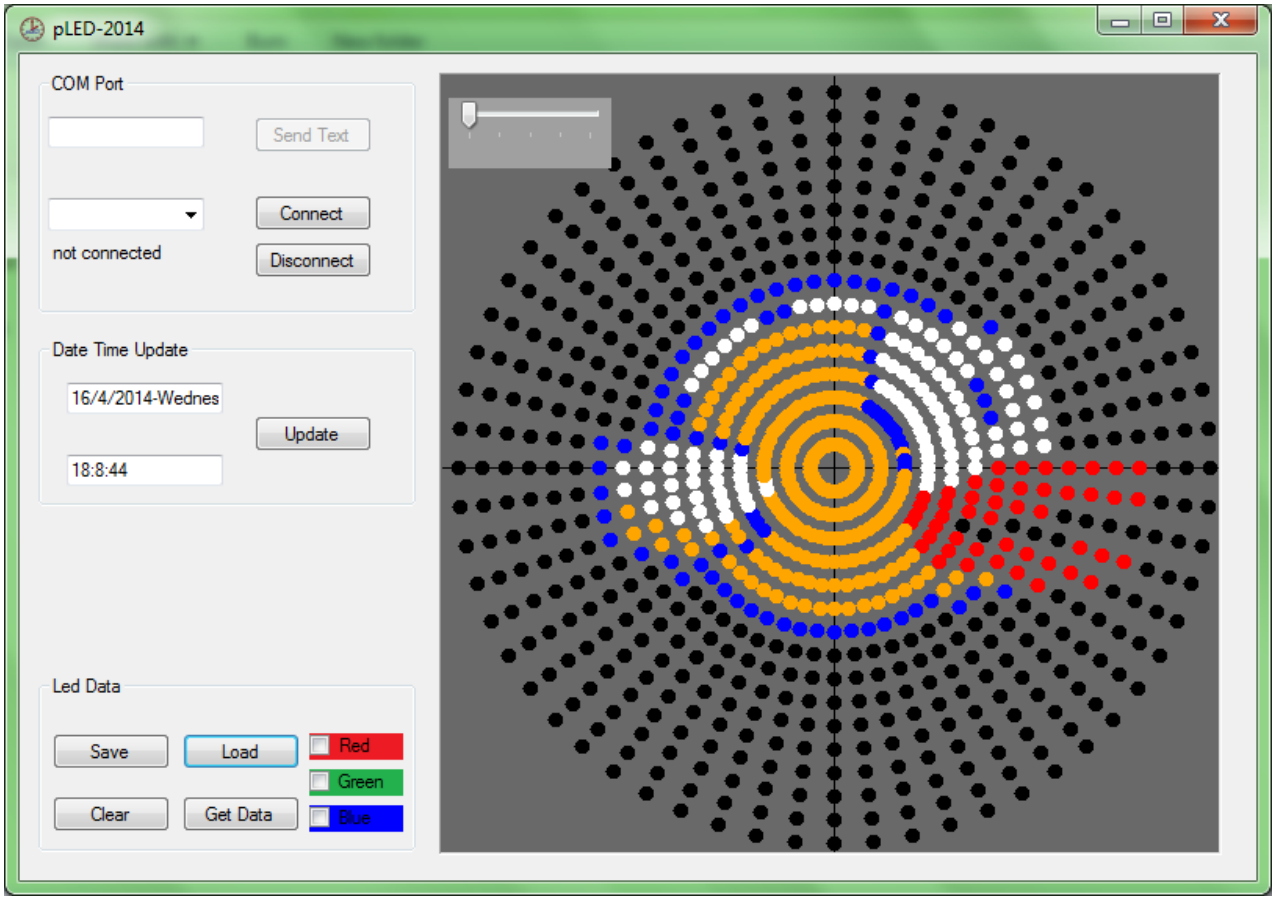


Figure 31: Configure tool interface

2. Testing

2.1 Scope of Testing

The test procedure with be split into 5 parts:

- Electrical connection on circuit board.
- IC components functionality
- Motor base's stability
- Circuit board's soldering
- Firmware

2.2 Plan of test

	Activities/ Tasks	Person in charge	Status	Planned Start date	Planned End date	Planned Effort (man-hours)
Initiation						
Study system						
	Research datasheet, PCB, SCH	BachDX, BanDX,LongNT, HaiDQ	Finished	23-Jan-14	24-Jan-14	**20**
	Learn how to use Digital Multimeter	BachDX, BanDX,LongNT, HaiDQ	Finished	24-Jan-14	25-Jan-14	**20**
Planning						
Test Plan						
	Create Schedule Test	BachDX	Finished	26-Jan-14	27-Jan-14	8
Design						
Create Testcase						
	Before Solder	BachDX	Finished	13-Feb-14	13-Feb-14	4
	After Solder	BachDX	Finished	14-Feb-14	15-Feb-14	8
	After Programing	BachDX	Finished	15-Feb-14	16-Feb-14	8
Execute Manual Test						
Dual colors led v1.0						
	Before Solder	BachDX	Finished	20-Jan-14	20-Jan-14	4
	After Solder	TungNT	Finished	25-Feb-14	26-Feb-14	8

	After Programing	LongNT	Finished	27-Feb-14	05-Mar-14	8
Triple colors led v1.0						
	Before Solder	BanDV	Finished	20-Jan-14	20-Jan-14	4
	After Solder	TungNT	Finished	25-Feb-14	26-Feb-14	8
	After Programing	HaiDQ	Finished	27-Feb-14	05-Mar-14	8
Triple colors led v2.0						
	Before Solder	BachDX	Finished	05-Mar-14	05-Mar-14	4
	After Solder	BachDX	Finished	10-Mar-14	10-Mar-14	8
	After Programing	LongNT	Finished	15-Mar-14	25-Mar-14	8
Test Report						
Dual colors led v1.0						
	Create Test Report	BachDX	Finished	05-Apr-14	05-Apr-14	4
Triple colors led v1.0						
	Create Test Report	BachDX	Finished	06-Apr-14	06-Apr-14	4
Triple colors led v2.0						
	Create Test Report	BachDX	Finished	07-Apr-14	07-Apr-14	4

2.3 Requirements for Testing

2.3.1 Test Items

- Voltmeter
- Microcontroller Debugger
- Windows laptop
- Human

2.3.2 Acceptance Test Criteria

- Successfully operate in 30 minutes without any harm to audiences.
- Receive and response user's input in less than 200ms
- Display correct real-time after power up.
- The system is robust and solid.

2.3.3 Testing Risks

- SMD components are too small to be tested.
- Conflict between hardware and software.
- Do not have much experience about electronic and hardware.
- Some components are made in china do not have datasheet.

2.4 Test Strategies

Bottom-Up Testing: Test the modules at the lower levels in the hierarchy, and then working up the hierarchy of modules until the final module is tested. This type of testing is appropriate for **object-oriented systems** in that individual objects may be tested using their own test drivers. They are then integrated and the object collection is tested.

2.4.1 Test Policies

- To effectively and efficiently provide timely, accurate, and useful quality risk management information.
- Test base on schematic and datasheet

- Quality risks are identified, and risk items are assessed to determine their level of risk. The level of risk determines test effort and test sequencing. Test results are reported in terms of mitigated and unmitigated risks.
- Test process:

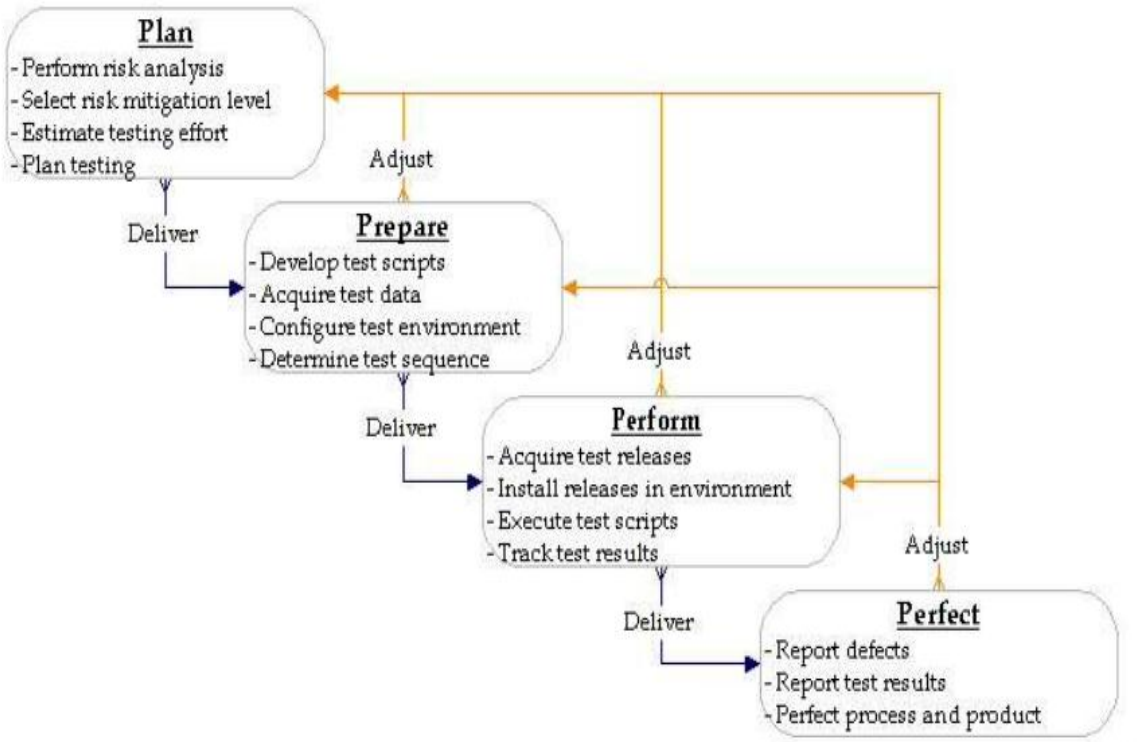


Figure 32: Test Process

2.4.2 Types of Testing

- Unit testing.
- Functional testing.
- Integration testing.
- System testing.
- Acceptance testing.
- Usability testing.
- Security testing.

2.4.3 Tools/Environments

Type	Purpose	Owner
Personal Computer	See PCB,SCH, Design	Tester
Digital Multimeter DT9205 A+	Check connection between components on circuit board	Tester
Digital Multimeter VC9803A+	Check connection between components on circuit board	Tester
Pickit 2	Download programing	Tester
Supply power 12V-1A	Supply power	Tester
Supply power 5V-1A	Supply power	Tester
Supply power 12V-2A	Supply power	Tester

3. Test Cases

3.1 Circuit board connection

3.1.1 Dual colors led v1.0

a) Before Solder

TestCase ID	Item Check	Description	Expectation	Status	Person Test	Test Date	Note
pLED_Dual_001	Hall	Hall has 3 PINs : VCC, GND, Output 1.Connect VCC hall with anode of supply power DC 3V 2.Connect GND hall with cathode of supply power DC 3V 3.Connect Output hall with anode of Led 4.Use magnet near hall	Led ON Hall work OK	OK	BachDX	1/20/2014	
pLED_Dual_002	RF Module	RF has 7 PINs: 4 PIN data, VCC, GND Connect RF with board, using code to check RF is available	4 functions are available	OK	BachDX	1/20/2014	
pLED_Dual_003	EEPROM	Check PIN available	8 PINs are intact	OK	BachDX	1/20/2014	
pLED_Dual_004	PIC 16F887	Check PIN available	44 PINs are intact	OK	BachDX	1/20/2014	

pLED_Dual_005	LED RBG	Connect anode and cathode of Led with supply power DC 3V	Led Blue, Green, Red ON	OK	BachDX	1/20/2014	
pLED_Dual_006	DS 1307	Check PIN available	8 PINs are intact	OK	BachDX	1/20/2014	
pLED_Dual_007	ISCP	Check PIN available	8 PINs are intact	OK	BachDX	1/20/2014	
pLED_Dual_008	Shift Register 74HC595	Check PIN available	16 PIN intact	OK	BachDX	1/20/2014	

b) After solder

TestCase ID	Item Check	Description	Expectation	Status	Person Test	Test Date	Note
pLED_Dual_009	Hall Sensor	Check connection between components on circuit board	Connect	OK	TungNT	2/25/2014	
pLED_Dual_010	RF Module	Check connection between components on circuit board	Connect	OK	TungNT	2/25/2014	
pLED_Dual_011	EEPROM	Check connection between components on circuit board	Connect	NG	TungNT	2/25/2014	
pLED_Dual_012	PIC 16F887	Check connection between components on circuit board	Connect	OK	TungNT	2/25/2014	

pLED_Dual_013	LED RBG	Check connection between components on circuit board	Connect	NG	TungNT	2/25/2014	11/16 Led Connect
pLED_Dual_014	DS 1307	Check connection between components on circuit board	Connect	NG	TungNT	2/25/2014	
pLED_Dual_015	ISCP	Check connection between components on circuit board	Connect	OK	TungNT	2/25/2014	
pLED_Dual_016	Shift Register 74HC595	Check connection between components on circuit board	Connect	OK	TungNT	2/25/2014	

c) After programing

TestCase ID	Item Check	Description	Expectation	Status	Person Test	Test Date	Note
pLED_Dual_0017	Hall Sensor	Check it using code test	Action	OK	LongNT	2/27/2014	
pLED_Dual_0018	RF Module	Check it using code test	Action	OK	LongNT	2/27/2014	
pLED_Dual_0019	EEPROM	Check it using code test	Action	NG	LongNT	2/27/2014	Fail after solder
pLED_Dual_0020	PIC 16F887	Check it using code test	Action	OK	LongNT	2/28/2014	
pLED_Dual_0021	LED RBG	Check it using code test	Action	NG	LongNT	3/1/2014	Fail after solder

pLED_Dual_0022	DS 1307	Check it using code test	Action	NG	LongNT	3/2/2014	Fail after solder
pLED_Dual_0023	ISCP	Check it using code test	Action	OK	LongNT	3/3/2014	
pLED_Dual_0024	Shift Register 74HC595	Check it using code test	Action	OK	LongNT	3/5/2014	

3.1.2 Triple colors led v1.0

a) Before Solder

TestCase ID	Item Check	Description	Expectation	Status	Person Test	Test Date	Note
pLED_Triple_001	Hall Sensor	Hall has 3 PINs : VCC, GND, Output 1.Connect VCC hall with anode of supply power DC 3V 2.Connect GND hall with cathode of supply power DC 3V 3.Connect Output hall with anode of Led 4.Use magnet near hall	Led ON Hall work OK	OK	BachDX	1/20/2014	
pLED_Triple_002	RF Module	RF has 7 PINs: 4 PIN data, VCC, GND Connect RF with board, using code to check RF is available	4 functions are available	OK	BachDX	1/20/2014	

pLED_Triple_003	EEPROM	Check PIN available	8 PINs are intact	OK	BachDX	1/20/2014	
pLED_Triple_004	PIC 16F887	Check PIN available	44 PINs are intact	OK	BachDX	1/20/2014	
pLED_Triple_005	LED RBG	Connect anode and cathode of Led with supply power DC 3V	Led Blue, Green, Red ON	OK	BachDX	1/20/2014	
pLED_Triple_006	DS 1307	Check PIN available	8 PINs are intact	OK	BachDX	1/20/2014	
pLED_Triple_007	ISCP	Check PIN available	8 PINs are intact	OK	BachDX	1/20/2014	
pLED_Triple_008	Shift Register 74HC595	Check PIN available	16 PIN intact	OK	BachDX	1/20/2014	

b) After Solder

TestCase ID	Item Check	Description	Expectation	Status	Person Test	Test Date	Note
pLED_Triple_009	Hall Sensor	Check connection between components on circuit board	Connect	OK	TungNT	2/25/2014	
pLED_Triple_010	RF Module	Check connection between components on circuit board	Connect	OK	TungNT	2/25/2014	
pLED_Triple_011	EEPROM	Check connection between components on circuit board	Connect	OK	TungNT	2/25/2014	

pLED_Triple_012	PIC 16F887	Check connection between components on circuit board	Connect	OK	TungNT	2/25/2014	
pLED_Triple_013	LED RBG	Check connection between components on circuit board	Connect	OK	TungNT	2/25/2014	
pLED_Triple_014	DS 1307	Check connection between components on circuit board	Connect	OK	TungNT	2/25/2014	
pLED_Triple_015	ISCP	Check connection between components on circuit board	Connect	OK	TungNT	2/25/2014	
pLED_Triple_016	Shift Register 74HC595	Check connection between components on circuit board	Connect	OK	TungNT	2/25/2014	

c) After programing

TestCase ID	Item Check	Description	Expectation	Status	Person Test	Test Date	Note
pLED_Triple_017	Hall Sensor	Check it using code test	Action	OK	LongNT	2/27/2014	
pLED_Triple_018	RF Module	Check it using code test	Action	OK	LongNT	2/27/2014	

pLED_Triple_019	EEPROM	Check it using code test	Action	OK	LongNT	2/27/2014	
pLED_Triple_020	PIC 16F887	Check it using code test	Action	NG	LongNT	2/28/2014	Cannot Programing
pLED_Triple_021	LED RBG	Check it using code test	Action	OK	LongNT	3/1/2014	
pLED_Triple_022	DS 1307	Check it using code test	Action	OK	LongNT	3/2/2014	
pLED_Triple_023	ISCP	Check it using code test	Action	OK	LongNT	3/3/2014	
pLED_Triple_024	Shift Register 74HC595	Check it using code test	Action	OK	LongNT	3/5/2014	

3.1.3 Triple colors led v2.0

a) Before Solder

TestCase ID	Item Check	Description	Expectation	Status	Person Test	Test Date	Note
pLED_Triple_001	Hall Sensor	Hall has 3 PINs : VCC, GND, Output 1.Connect VCC hall with anode of supply power DC 3V 2.Connect GND hall with cathode of supply power DC 3V 3.Connect Output hall with anode of Led 4.Use magnet near hall	Led ON Hall work OK	OK	BachDX	3/5/2014	
pLED_Triple_002	RF Module	RF has 7 PINs: 4 PIN data, VCC, GND Connect RF with board, using code to check RF is available	4 functions are available	OK	BachDX	3/5/2014	
pLED_Triple_003	EEPROM	Check PIN available	8 PINs are intact	OK	BachDX	3/5/2014	
pLED_Triple_004	PIC 16F887	Check PIN available	44 PINs are intact	OK	BachDX	3/5/2014	
pLED_Triple_005	LED RBG	Connect anode and cathode of Led with supply power DC 3V	Led Blue, Green, Red ON	OK	BachDX	3/5/2014	
pLED_Triple_006	DS 1307	Check PIN available	8 PINs are intact	OK	BachDX	3/5/2014	
pLED_Triple_007	ISCP	Check PIN available	8 PINs are	OK	BachDX	3/5/2014	

			intact				
pLED_Triple_008	Shift Register 74HC595	Check PIN available	16 PIN intact	OK	BachDX	3/5/2014	

b) After Solder

TestCase ID	Item Check	Description	Expectation	Status	Person Test	Test Date	Note
pLED_Triple_009	Hall Sensor	Check connection between components on circuit board	Connect	OK	TungNT	3/10/2014	
pLED_Triple_010	RF Module	Check connection between components on circuit board	Connect	OK	TungNT	3/10/2014	
pLED_Triple_011	EEPROM	Check connection between components on circuit board	Connect	OK	TungNT	3/10/2014	
pLED_Triple_012	PIC 16F887	Check connection between components on circuit board	Connect	OK	TungNT	3/10/2014	
pLED_Triple_013	LED RBG	Check connection between components on circuit board	Connect	OK	TungNT	3/10/2014	
pLED_Triple_014	DS 1307	Check connection between components on circuit board	Connect	OK	TungNT	3/10/2014	

pLED_Triple_015	ISCP	Check connection between components on circuit board	Connect	OK	TungNT	3/10/2014	
pLED_Triple_016	Shift Register 74HC595	Check connection between components on circuit board	Connect	OK	TungNT	3/10/2014	

c) After programing

TestCase ID	Item Check	Description	Expectation	Status	Person Test	Test Date	Note
pLED_Triple_017	Hall Sensor	Check it using code test	LED toggle each time	OK	LongNT	3/15/2014	
pLED_Triple_018	RF Module	Check it using code test	Specific LED turn on for each button	OK	LongNT	3/16/2014	
pLED_Triple_019	EEPROM	Check it using code test	Action	OK	LongNT	3/17/2014	
pLED_Triple_020	PIC 16F887	Check it using code test	Response from debugger	OK	LongNT	3/18/2014	
pLED_Triple_021	LED RBG	Check it using code test	Action	OK	LongNT	3/19/2014	

pLED_Triple_022	DS 1307	Check it using code test	Led flashing once each second	OK	LongNT	3/20/2014	
pLED_Triple_023	ISCP	Check it using code test	Action	OK	LongNT	3/21/2014	
pLED_Triple_024	Shift Register 74HC595	Check it using code test	Action	OK	LongNT	3/25/2014	

d) Functional test

TestCase ID	Description	Expectation	Status	Person Test	Test Date	Note
pLED_Triple_024	User press button "A" on RF control	Display Analog Clock	OK	BachDX	3/27/2014	
pLED_Triple_025		Display Real Time	OK	BachDX	3/27/2014	
pLED_Triple_026		Matching Image Design	OK	BachDX	3/27/2014	
pLED_Triple_027		Clock display smoothly	N/A	BachDX	3/27/2014	Interrupt overlap
pLED_Triple_028	User press button "B" on RF control	Display Digital Clock	OK	BachDX	3/27/2014	
pLED_Triple_029		Display Real Time	OK	BachDX	3/27/2014	
pLED_Triple_030		Matching Image Design	OK	BachDX	3/27/2014	
pLED_Triple_031		Clock display smoothly	N/A	BachDX	3/27/2014	Interrupt overlap
pLED_Triple_032	User press button "C" on RF control	Display Image	OK	BachDX	3/27/2014	
pLED_Triple_033		Matching Image	OK	BachDX	3/27/2014	

		Design				
pLED_Triple_034		Image display smoothly	N/A	BachDX	3/27/2014	Interrupt overlap
pLED_Triple_035	User press button "D" on RF control	Display Text	OK	BachDX	3/27/2014	
pLED_Triple_036		Matching Image Design	OK	BachDX	3/27/2014	
pLED_Triple_037		Text display smoothly	N/A	BachDX	3/27/2014	Interrupt overlap

Interrupt overlap: In PIC, if multiple interrupt happen at the same time, only 1 interrupt can be process at a time. In a time critical system like pLED, a small delay of each interrupt will make a big impact to visual displayed (Which is the shuttering in finished product)

3.2 Base's stability

TestCase ID	Item Check	Description	Expectation	Status	Person Test	Test Date	Note
pLED-001	Motor	Using supply power 5V-2A	Stable	OK	BanDV	3/25/2014	
pLED-002	Base	Check it stable while motor running	Stable	OK	BachDX	3/25/2014	
pLED-003	Circuit Board	Check it balance	Stable	OK	TungNT	3/25/2014	

4. Test Logs

4.1 Defect Logs

Defect Log								
Defect Id	**Component Name**	**Test Case ID**	**Defect Description**	**Date Raised**	**Serverity**	**Status**	**Closed Date**	**Comments**
Dual colors LED v1.0								
1	Hardware	pLED_Dual_011	Not correct with circuit principle	2/25/2014	High	Closed	2/26/2014	Item dropped
2	Hardware	pLED_Dual_013	Not correct with circuit principle	2/25/2014	High	Closed	2/26/2014	LEDs are blown out
3	Hardware	pLED_Dual_014	Not correct with circuit principle	2/25/2014	High	Closed	2/26/2014	Item dropped
Triple colors LED v1.0								
4	Hardware	pLED_Triple_020	PIC not programing	2/28/2014	High	Closed	2/28/2014	PIC is blown
Triple colors LED v2.0								
5	Sofware	pLED_Triple_027	Clock display smoothly	3/27/2014	High	Fixed	3/29/2014	
6	Sofware	pLED_Triple_031	Clock display smoothly	3/27/2014	High	Fixed	3/31/2014	
7	Sofware	pLED_Triple_034	Image display smoothly	3/27/2014	High	Fixed	04/02/2014	
8	Sofware	pLED_Triple_037	Text display smoothly	3/27/2014	High	Fixed	04/04/2014	

4.2 Test Reports

Project Name	Propeller LED	**Creator**	BachDX
Project Code	pLED	**Reviewer/Approver**	TungNT
Document Code		**Issue Date**	April 04th 2014

No	Version of product	Pass	Fail	Untested	N/A	Number of test cases
1	Dual colors LED v1.0	21	3	0	0	24
2	Tripple colors LED v1.0	23	1	0	0	24
3	Tripple colors LED v2.0	20	4	0	0	24

CHAPTER 6: USER'S GUIDE

1. Purpose

This part of the document is to tell user how to use the propeller LED, and some notice on using that may help improve life of product.

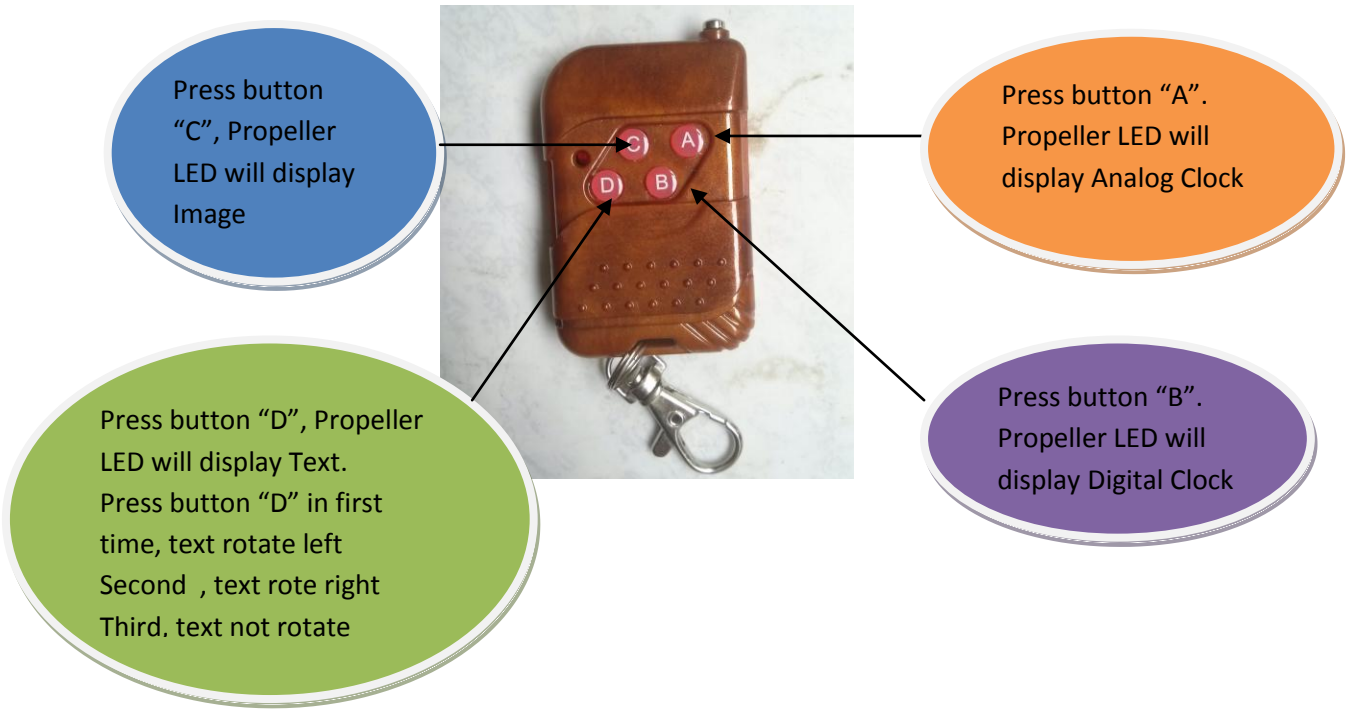
2. Function's Description

Product has 5 functions:

- Display Analog Clock
- Display Digital Clock
- Display Text
- Display Image
- Setup Time

3. Detailed Guidelines

3.1 Using Remote Control



3.2 Using application setup time

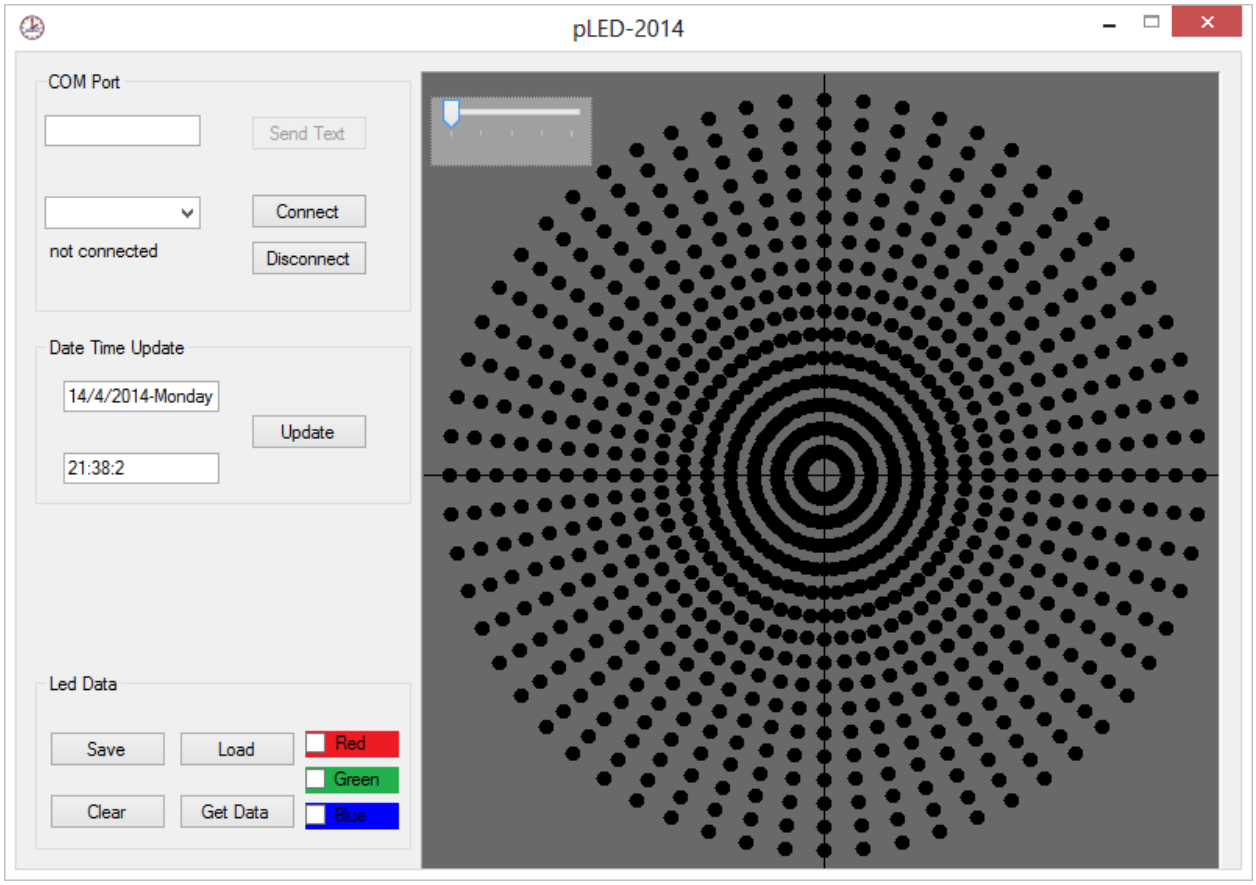


Figure 33: Time update Software

Connect pLED with PC thought RS232. Click button "Connect" then click button "Update".

CONCLUSION

1. Development

Now the propeller only have 16 LEDs, this number limit the display ability of the propeller. In the future, we want to develop this product using more LED so the display can be improved a lot. One more function we want to develop is a tool that can convert image from any extension to data which can be loaded to the propeller directly.

2. Advantage and Disadvantage

2.1 Advantage

- Have already learnt about Micro Controller and another IC
- Ability to use new technology quickly
- Have experienced supervisor

2.2 Disadvantage

- Inexperience of electronic component's error and replacement
- Inexperience of producing mechanical component
- Inexperience of design circuit board
- Have to outsource Hardware (produce and solder circuit board)

3. Skill learned

- Design Schematic follow requirement
- Design printed circuit board, order board, solder IC component
- Design hardware and calculator stable for system
- Planning task, create schedule
- Working in group, brainstorming, find problem and resolve problem
- Working with version control software

REFERENCE

[1] http://www.bobblick.com/techref/projects/propclock/propclock.html

[2] http://avclock.com/xemchude/1/huong-dan-thiet-ke-mach-nguyen-ly-propeller-clock-dong-ho-led-quay.html

[3] http://en.wikipedia.org/wiki/Embedded_system#Interrupt-controlled_system

[4] https://www.ccsinfo.com/forum/viewtopic.php?t=23255

[5] PIC16F887 datasheet
http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en026561

[6] DS1307 datasheet

http://datasheets.maximintegrated.com/en/ds/DS1307.pdf

[7] 74HC595 datasheet

http://www.nxp.com/documents/data_sheet/74HC_HCT595.pdf