Light Control Systems for Smart Homes

Maphothoane Lebohang Phillip

NATIONAL UNIVERSITY OF LESOTHO

Submitted to the Department of Physics and Electronics

In Partial Fulfilment of Bachelors Degree

in Electronics Engineering

Under the Supervision of Mr.Mokoena

May 2010

Abstract

Energy is very important and necessary to every day life. Increasing fuel costs and high demand for electricity which cause load shading has made energy conservation to become an absolute necessity. The aim of the project is to design light control system for smart homes that helps to conserve energy by controlling light in the house. The system is operated manually and automatically. The system is operated automatically with the help of sensors, sensing the luminosity inside and outside the house.

The importance of automatic operation is to try to solve the problem of having the house lights off during the night if there is nobody around to turn the house lights on. During the night, the system automatically close the blinds and turn the lights on, and turn them off after a period of time that the owner of the house would require. At day light, the system automatically open the blinds and turn the lights off. The blinds are opened to a certain angle depending on the amount of light needed in the house or closed if there is enough light in the room. The system also have interface for the computer and on the interface the user can manually control all the lights in the house.

When building the system, components were tested to verify whether they are in good condition. Supporting circuits such as Light detector circuit and Infrared Photo Detector (IPD) circuit were constructed and checked for reliability in hot and cold weather conditions. Circuit diagram for turning Unipolar Stepper Motor (USM) was constructed and tested. Significant results obtained are:

- Different pulse sequence of 1 and 0 used to turn USM in different directions.
- Output voltage of Light Dependent Resistor (LDR) in day light whereby its value is used in the Light detector code to note that it is bright, hence the lights must turn off and output voltage of LDR in dark shade whereby its value is used to note that it is dark, therefore the lights must turn on.

- Output voltage of IPD when sensing motion, whereby its value is used in a complete code of automatic light control system to node that there is motion. Therefore, lights must be on depending on the amount of light that is available in the house.

Contents

Acknowledgements

I would like to express my sincere gratitude to Mr Mokoena, my supervisor and Mr Mathaba, project co−supervisor for their support and their valuable guidance. I would like to thank all the staff of Department of Physics and Electronics mostly Ntate Matobo, for helping with project Equipment.

I would also like to thank all my classmates, especially Nchaba, for helping with latex instructions. Limpho and Makhesi for helping with java language. Khabo, Sehlabaka, Mobe and Makhele for supporting me through hard times.

I would also like to thank Mr Seforo, Dr Makhele and Dr Thamae for thier valuable questions during presentations, I learned a lot from those questions.

I am extremely grateful to Dr Moeketsi Mpholo, for that pressure he put on us, because of that pressure I was able to finish the project in time.

List of Abbreviations

ADC	-	Analog to Digital Converter
AST	-	Asynchronous Serial Transmission
BSM	-	Bipolar Stepper Motor
CFL	-	Compact Fluorescent Lamps
CPU	-	Central Processing Unit
CREN	-	Continuous Receive Enable
CTS	-	Clear to Send
DCE	-	Data Communication Equipment
DTE	-	Data Terminal Equipment
EIA	-	Electronic Industries Association
GIMD	-	General Instrument's Microelectronics Division
GLS	-	Incandescent Lamps
GUI	-	Graphical User Interfce
IDE	-	Integrated Development Environment
IPD	-	Infrared Photo Detector
JDK	-	Java Development Kit
JFET	-	Junction Field Effect Transistor
LDR	-	Light Dependent Resistor
LED	-	Light Emitting Diode
MCU	-	Microcontroller
PC	-	Personal Computer
PIC	-	Peripheral Interface Controller
RAM	-	Random Access Memory
RCIF	-	Receive Interrupt Flag
RCSTA	-	Receive Status and Control Register

RTS	-	Ready to send
SHA	-	Smart Homes Association
SPEN	-	Serial Port Enable
SST	-	Synchronous Serial Transmission
TIA	-	Telecommunications Industry Association
TTL	-	TransistorTransistor Logic
TxD	-	Transmit Data
USART	-	Universal Synchronous Asynchronous Receiver Transmitter
USM	-	Unipolar Stepper Motor
Vpp	-	Programming Voltage

List of Figures

List of Tables

Chapter 1

Introduction

The terms smart homes, intelligent homes, home networking have been used for more than a decade to introduce the concept of networking devices and equipment in the house. According to Smart Homes Association (SHA) the best definition of smart home technology is the integration of technology and services through home networking for a better quality of living. The basic idea of smart home is to employ sensors and control systems to monitor a dwelling, and accordingly adjust the various mechanisms that provide heat, ventilation, lighting, and other services. By more closely tuning the dwellings mechanical systems to the dwellers needs, the automated intelligent home can provide a safer, more comfortable, and more economical dwelling. For example, the electronic controller of an automated home can determine when the dwellers have gone to bed and turn off the lights [1].

Light control system for smart homes can enable disabled and elderly people to lead safe and independent lives in their own homes by providing an environment that is constantly monitored to ensure that the householder is safe and automate specific tasks that a householder is reluctant to perform like turning lights on or off [2]. Light control system for smart homes, can also provide a safe and secure environment. For instance, when the owner of the house is not around the blinds can automatically open during the day and close at night. This would provide security since the buglers would think that the owner of the house is still around. Moreover, the blinds can automatically open to a desired angle with the help of light sensors whereby the light sensors detect how much light is needed in the house, hence saving energy because there would be no need for the lights to be on.

In this project I used LDR sensors to measure light intensity and IPD to detect any motion in the house. I used USM to open and close the blinds depending on the amount of light needed in the house. The operation of sensors and USM are controlled by the microcontroller. I used PIC16F887 to control the input data from the sensors and also to provide with necessary output to the USM, and the Light Emitting Diodes (LEDs) which represents lighting bulbs. Since the system has to allow the user to control it manually using computer, I used RS232 serial cable to connect Personal Computer (PC) to the circuit board.

Chapter 2 will discuss about the design specification, Chapter 3 will talk about concept selection, while Chapter 4 will talk about literature review (details of sensors, microcontrollers, motors, rs 232 cable and LEDs), Chapter 5 will discuss implementation, program development and flow charts for the system and Chapter 6 will discuss about the results obtained. The last Chapter would be the conclusion of the report. Appendix and code for the entire system will be available at the end of the report followed by the bibliography.

Chapter 2

Design Specification

2.1 Foreword

Every commercial house, electricity bill always have cost from lighting. It is well known that geysers, heaters and electric stove consume more energy than other electric appliances, hence most people do not use them especially in poor countries. About 50% of energy cost in such poor countries are for lighting because lighting is used every day than other electric appliances.

The energy is wasted mostly in rooms that are unoccupied. Hence the system is to be designed in such a way that at least 40% of energy should be spent in lighting and this would be achieved by only lighting the rooms that are occupied. When it comes to offices, there is valuable energy that is wasted on corridor lights, hence the system is to be designed in such away that corridor lights would be on only if there is a person moving on the corridor.

2.2 Scope

The system would be successful only if the following task are carried:

- Studying Assembly language because most of microcontroller's data sheet have their instruction examples written in Assembly language. Also studying Java, C and C++ because it is easy to create computer interface using one of three languages.
- Proper installation of electrical wires.
- Studying circuit design and analysis so as to design hardware of the system.
- Implementing and testing of the system.
- Studying operation of stepper motor and motor drivers.
- Surveying various kinds of microcontrollers in order to select the most suitable one.

2.3 Performance Requirements

- The system should operate well in summer and in winter, meaning it must not be affected by increase or decrease in temperature.
- Humidity range is accepted to be from 0 to 80%.
- The system has to be user friendly, such that every one in the house can use it. Again the system should be operable in a sense that it can allow the user to actually control it.

- The system should be designed in such away that it can be easily upgraded.
- The cost of ownership should be less than cost of initial purchase.
- The system should be maintainable, so that the failed item should be restored to its satisfactory operational state.

2.4 Manufacture Requirements

The system requires a correct code written using a reliable programming language and be installed in a microcontroller. The system would be assembled manually and it would be assembled earlier before installation. Required materials to be assembled are micro-controller, sensors, motors, blinds, electrical wires and light bulbs. The system should be dissembled and reassembled easily for transportation. Time of installation of the system into the house should be less than a day, although it depends on the size of the house. Time of installation for a 5 room house is expected to be one day.

2.5 Acceptance Standards

The system would be tested before it leaves the assembly and it will also be tested during installation to the customer to confirm that no fault occurred during transportation because if there is a fault, it is likely that the system can cause a disaster. International standards (such as developing child resistant packages that offers an adequate physical barrier between a child and a range of hazardous products like electrical wire cables that are not insulated, liquid fuels and solvents) would be followed. Regulations and codes of practise would also be followed, such as codes that offers examples of good practise and that gives advice on ways in which a system can be carried out safely and efficiently. The system would not conflict with existing patents.

2.6 Operation Requirements

There must be no specific skills required from the operator or the user and the system must be user friendly. For maintenance, it is required that the user must call for help to any nearby electronic engineering store. For safety, there would be a guide book given to the user showing him or her how to optimally and efficiently use the system.

Chapter 3

Concept Selection

3.1 Possible Methods of Implementation

There are different approaches or possible methods of implementing the system and they are referred to as concepts. There are three possible concepts: Concept A, Concept B and Concept C, whose diagrams are shown in Figures 3.1, 3.2, and 3.3 respectively. The concepts are evaluated and compared to each other. The best concept with highest percentage as shown in Table 3.2 is chosen as the best concept.

3.1.1 Concept A

- It uses two sensors, motion sensor and the light sensor to automatically control the light in the house.
- It uses the interface from the computer such that the user can manually control the light in the house.

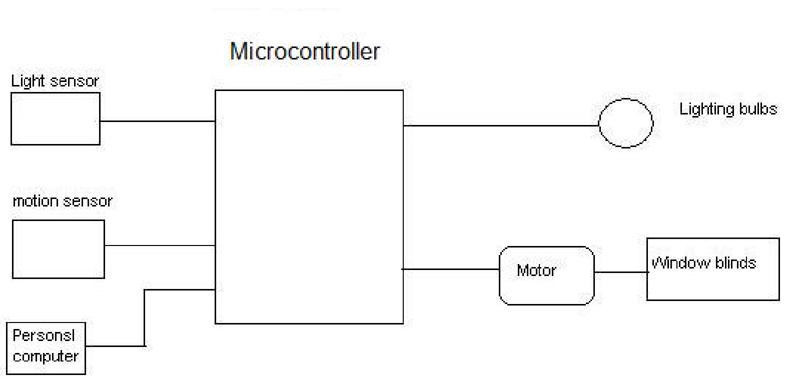


Figure 3.1: Concept A

3.1.2 Concept B

- It uses the remote control, where by the user can control the light in the house.

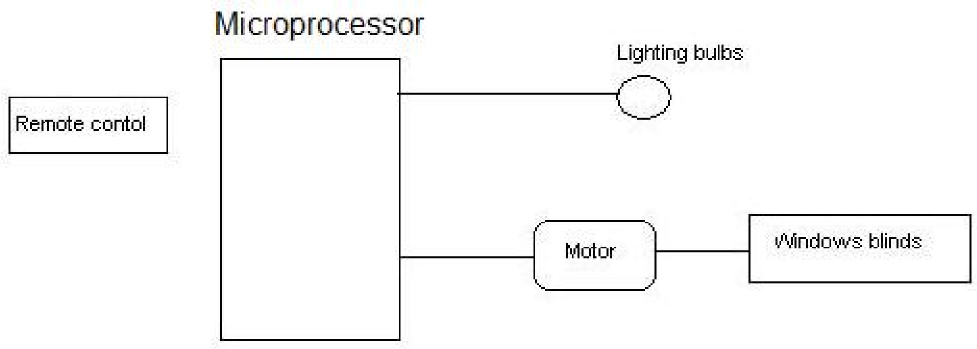


Figure 3.2: Concept B

3.1.3 Concept C

- It uses the time controller, whereby the light in the house is controlled by time. For instance, as sun rise the lights turn off and the blinds open while night the lights turn on and the blinds closes.

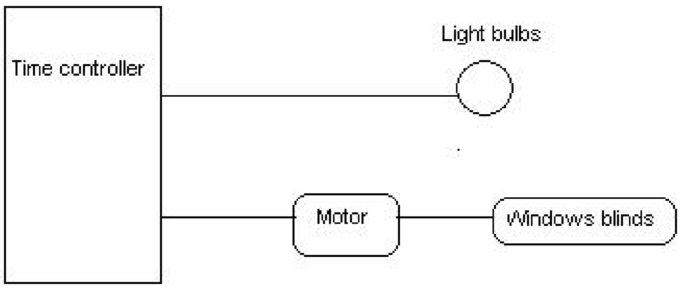


Figure 3.3: Concept C

3.2 Choice of Implementation scheme

Table 3.1 shows binary dominance matrix for the system whereby functions and constraints are ranked in order of relative importance. Only 105 individual decisions are made from 15 criteria which are ranked using the checking formula

$$0.5n(n-1) \tag{3.1}$$

Where n is the number of available criteria. Weighting factor for each criteria is found by dividing 105 into the individual total. Table 3.2 shows complete concept selection worksheet for the systems whereby criteria are compared with each potential solution. Each concept is scored as to how well it satisfy each criterion and is allocated a percentage. The percentages are multiplied by the weighting factor for that special criterion and the results are added to give total percentage [3].

Table 3.1: Binary dominance matrix for Light Control Systems for Smart Homes.

Criteria	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total	Weight
1 Manufacturing costs	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	13	0.124
2 Reliability	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	12	0.114
3 Maintenance	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	5	0.048
4 quality	0	0	1	0	1	1	0	1	1	0	1	1	0	0	0	7	0.067
5 Ergonomics	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	3	0.029
6 Ease of upgrading	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	4	0.038
7 Assembly	0	0	0	1	1	0	0	1	1	0	1	1	0	0	0	6	0.057
8 Noise	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0.01
9 Environmental cond.	0	0	1	0	1	0	0	1	0	0	1	1	0	0	0	5	0.048
10 Process	0	0	1	1	1	1	1	0	1	0	1	1	0	0	0	8	0.076
11 quantity	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	2	0.019
12 Installation	0	0	1	0	1	0	0	1	0	0	1	0	0	0	0	4	0.038
13 Material	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	10	0.095
14 Complexity	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	11	0.105
15 Safety	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	14	0.133
																105	1

Summarizing Table 3.2. Concept A tends to be the best concept because it has scored more points on Ergonomics, which means it can be designed for comfort, safety and productivity. Concept A can be easily upgraded, it can operate well in different environmental conditions, it can be easily maintained, it is reliable, it has high quality and moreover it is safer from causing short circuits or any kind of electrical shock.

The principal disadvantages of Concept B and Concept C is that, they consume more power, they are very expensive and they are bulky in size while Concept A has low power consumption and it is compact in size. The

Table 3.2: Complete concept selection worksheet for the systems.

		Concepts		
Criteria	weighting	A%	B%	C%
15 Safety	0.133	90	90	90
1 Manufacturing costs	0.124	60	60	70
2 Reliability	0.114	90	85	80
14 Complexity	0.105	80	75	70
13 Material	0.095	80	70	75
10 Process	0.076	80	75	70
4 quality	0.067	90	85	85
7 Assembly	0.057	80	75	80
9 Environmental conditions	0.048	90	90	80
3 Maintenance	0.048	80	90	75
12 Installation	0.038	70	80	70
6 Ease of upgrading	0.038	90	90	80
5 Ergonomics	0.029	90	80	80
11 quantity	0.019	70	80	70
8 Noise	0.01	60	60	60
		81.12	78.835	77.21

Best concept is Concept A with a score of 81.12%

fact that Concept B is microprocessor based, it has the following drawbacks:

- It would be difficult to upgrade the system because microprocessor has limitations on the size of data. Solution to the problem would be networking of many microprocessors, but this would make the system to be complex and bulky in size.
- Microprocessor does not support floating−point operations.

- Microprocessor over heat physically hence the system can not operate well in summer.
- Microprocessor is not bit addressable because it does not have in−built memory.
- Microprocessor instructions are usually more than 1 byte and it takes many machine cycles to execute an instruction. Therefore, the system would be very slow.

The fact that Concept A is microcontroller based, it has the following advantages:

- Microcontroller is able to control a variety of processes and devices independently (such as switches, buttons and sensors), therefore upgrading of the system can be easily carried out.
- Microcontroller has in−built memory Random Access Memory (RAM), hence it can executes instructions fast.
- Microcontroller has programmable serial ports, as the result, serial communication can be easily established.
- Microcontroller instructions are generally 1 byte and excite in one cycle. Therefore, the system would be be very fast.

Chapter 4

Literature Review

4.1 Sensors

A sensor is a device that detects or measure a physical quantity. Sensors convert physical variables to signal variables. Sensors are often referred to transducers because they are devices that convert input energy of one form into output energy of another form. Sensors can be categorized into two broad classes depending on how they interact with the environment they are measuring. They can be categorized as passive sensors and active sensors. Passive sensors do not add energy as part of the measurement process but may remove energy in their operation.

One example of a passive sensor is a thermocouple, which converts a physical temperature into a voltage signal. In this case, the temperature gradient in the environment generates a thermoelectric voltage that becomes the signal variable. Active sensors add energy to the measurement environment as part of the measurement process. An example of an active sensor is a radar or sonar system, where the distance to some object is measured by actively sending out a radio (radar) or acoustic (sonar) wave to reflect off some object and measure its range from the sensor. There are several types of sensors such as infrared sensors, motion sensors and light sensors [4].

4.1.1 Light Sensor

Light sensor is a mechanical or electronic apparatus that detects light. It is used to provide information on distance, shape, speed and dimensions. There are many different kinds of light sensors, my interest is on LDR. LDR can be referenced as a photoconductor or photoresistor because it is an electronic component whose resistance is inversely proportional to the intensity of incident light [5].

4.1.2 Motion sensor

Motion can be detected by measuring change in speed or vector of an object in a field of view. This can be achieved by mechanical devices that quantifies motion that can be either integrated with or connected to other devices that alert the user of the presence of a moving object within the field of view [6]. IPD (rs:[1] 2017729) is one type of motion sensor I am using in this project.

4.2 Stepper Motor

Stepper motor is an electromechanical device which converts electrical pulses into discrete mechanical movements. The shaft of a stepper motor rotates in

[1]Largest electronic, electrical and industrial products catalogue and data library online in Asia offering secure online ordering with same day despatch.

discrete steps increments when proper sequence of electrical command pulses are applied to it. There are three types of stepper motor; variable reluctance, permanent magnet and hybrid. They differ in terms of construction based on the use of permanent magnets and iron rotors with laminated steel stators [7].

4.2.1 Unipolar Stepper Motor

The USM has five or six wires and four coils. The center connections of the coils are tied together and used as the power connection. Figure 4.1 shows the USM with a center tap on each of two windings. To rotate the

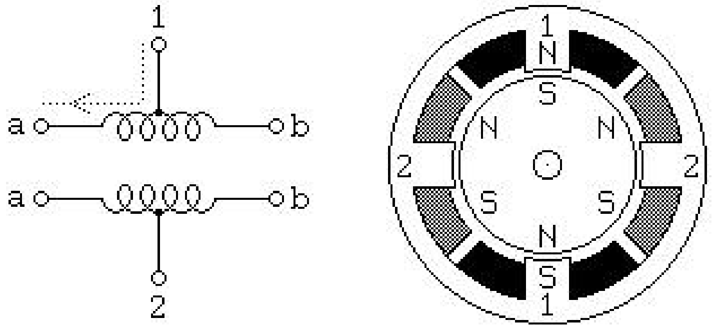


Figure 4.1: Unipolar stepping motor[8]

motor, pulse sequence or power is applied to the two windings in sequence. Assuming positive logic, where a 1 means turning on the current through a motor winding. The normal stepping sequence for four-coil unipolar steppers is shown below.

Winding 1a = 10001

Winding 1b = 00100

Winding 2a = 01000
Winding 2b = 00010 [8]

4.2.2 Bipolar Stepper Motor

Bipolar Stepper Motor (BSM) have two coils which are controlled by changing the direction of the current flow through the coils in the proper sequence. Driving a USM is similar to driving a BSM and the same pulse sequence is

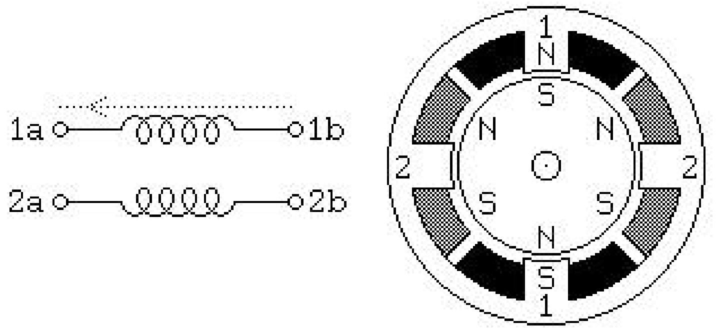


Figure 4.2: Bipolar stepper motor[8]

used. The only difference between driving a USM and driving a BSM is that there is an extra wire in a USM called center tap and BSM does not have center tap as it can be seen in Figure 4.2. Therefore the circuit of driving USM differs from the circuit used to drive BSM.

4.2.3 Motor driver

There are many types of motor drivers. In this project I chose to use ULN2803AN because it can operate at high voltage and high current. It is usually referred to high-current Darlington transistor array. It is made up of eight npn Darlington pairs which have high-voltage outputs with common-cathode clamp diodes for switching inductive loads. Current rating of each Darlington pair is 500mA [9].

4.3 Microcontrollers

A microcontroller is defined as computer on a chip or a single-chip computer because it contains memory and input-output interfaces in addition to the Central Processing Unit (CPU). Micro suggests that the device is small, and controller implies that the device control objects, processes, or events. In other words, a microcontroller can be described as an embedded controller, because the microcontroller and its support circuits are often built into, or embedded in the devices they control.

A microcontroller is similar to the microprocessor inside a personal computer, whereby microprocessor consist of arithmetic and logic devices having inputs and output operations, and a software programmed controller. Both microprocessors and microcontrollers contain a central processing unit which executes instructions that perform the basic logic, math and data-moving functions of a computer. To make a complete computer, microprocessor requires memory for storing data, programs and input or output (I/O) interfaces for connecting external devices like keyboards and displays [10].

4.3.1 PIC16F887

Peripheral Interface Controller (PIC) is a family of Harvard architecture microcontrollers made by Microchip Technology with built-in RAM, memory, internal bus, and peripherals that can be used for many applications. PICs are derived from the PIC1640 originally developed by GIMD. There are different types of PICs classified up into two major categories which are 8 bit microcontrollers and 16 bit microcontrollers. Table 4.1 shows subdivision of each category.

8-bit MCU Product family	16-bit MCU Product Family
PIC10	PIC24FData carrier detect
PIC12	PIC24H
PIC14	dsPIC30
PIC16	dsPIC33
PIC18	

Table 4.1: MCU product family.

PIC microcontrollers in the PIC16 and PIC18 families are considered mid-level microcontrollers while 16-bit PICs are considered high-end microcontrollers opposed to microcontrollers in the PIC10 through PIC14 families which are considered low-end microcontrollers [11]. Figure 4.3 shows the pin diagram for PIC16F887.

4.4 RS232

In serial communication, transmission of data between a computer and a device like programable instrument or another computer is usually done by a

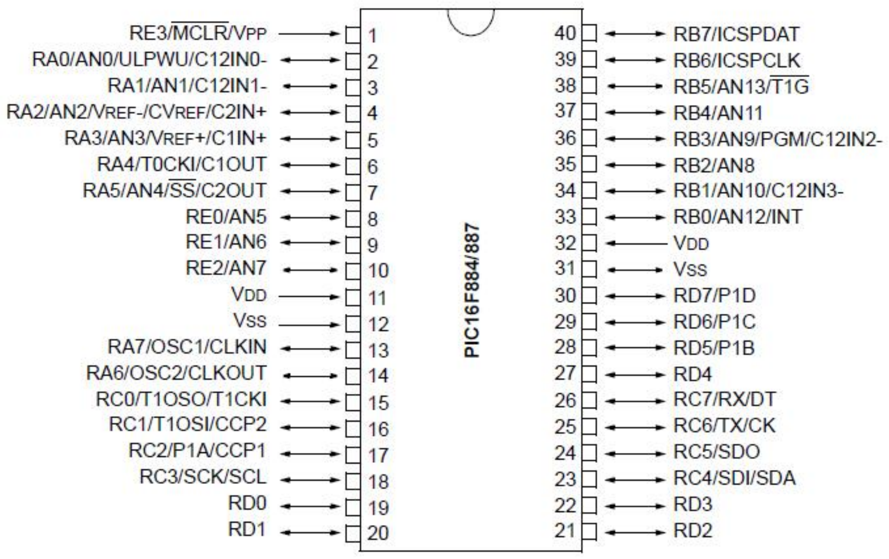


Figure 4.3: 40 pin diagram for PIC16F887 [12]

cable called RS232 or species of serial connection described in a specification written by the EIA in conjunction with TIA specifying the serial interface between Data Terminal Equipment (DTE) and Data Communication Equipment (DCE). The RS232 standard includes electrical signal characteristics, interface mechanical characteristics and functional description of interchange circuits [13] .

4.4.1 25-pin D-type connector

RS-232 standard states that DTE devices use a 25-pin male connector, and DCE devices use a 25-pin female connector. In DB-25 connector most of the pins are not needed for normal PC communications and mostly used pins are 22. Table 4.2 shows 25-pin D-type connector pin assignment [13] .

4.4.2 9-pin D-type connector

The introduction of smaller serial ports led to 9 pins and new PCs are equipped with male D type connectors having only 9 pins. 9-pin D-type connector pin assignment is shown in Table 4.3.

4.5 Light Emitting Diode

LED is a semiconductor device which converts electricity into light. LED is based on the semiconductor diode. The electrons recombine with holes within the devise when a diode is forward biased, releasing energy in a form of photons. The color of light corresponding to the energy of the photon is determined by energy gap of the semiconductor [14].

4.6 USART

Universal Synchronous Asynchronous Receiver Transmitter (USART) can be configured as a full duplex asynchronous system that can communicate with peripheral devices such as computers and it can also be configured as a half duplex synchronous system that can communicate with peripheral devices such as analog to digital or digital to analog integrated circuits. There are two primary forms of serial transmission which are Synchronous Serial Transmission (SST) and Asynchronous Serial Transmission (AST).

SST requires the sender and receiver to share a clock or the sender must provide a timing signal so that the receiver knows when to read the next bit of data. While AST requires the sender to send data to the receiver without sending clock signal, but the sender and the receiver must agree on timing parameters in advance[15].

Pin Number	Signal	Description
1	PG	Protective ground
2	TD	Transmitted data
3	RD	Received data
4	RTS	Request to send
5	CTS	Clear to send
6	DSR	Data set ready
7	SG	Signal Ground
8	CD	Carrier detect
9	+	Voltage (testing)
10	-	Voltage (testing)
11		
12	SCD	Secondary CD
13	SCS	Secondary CTS5
14	STD	Secondary TD
15	TC	Transmit Clock
16	SRD	Secondary RD
17	RS	Receiver clock
18		Ready to Send
19	SRS	Secondary RTS
20	DTR	Data Terminal Ready
21	SQD	Signal Quality Detector
22	RI	Ring Indicator
23	DRS	Data rate select
24	XTC	External Clock
25		

Table 4.2: 25-pin D-type connector pin assignment

Pin Number	Signal	Description
1	DCD	Data carrier detect
2	RxD	Receive Data
3	TxD	Transmit Data
4	DTR	Data terminal ready
5	GND	Signal ground
6	DSR	Data set ready
7	RTS	Ready to send
8	CTS	Clear to send
9	RI	Ring Indicator

Table 4.3: 9-pin D-type connector pin assignment

Chapter 5

Design and Implementation

5.1 System Operation

The fully functional system should perform the following operations:

- Perceive the existence of human moving in the room or detect if the room is occupied or not.
- keep the lights off and the blinds closed if the room is unoccupied especially during the midnight.
- If there is sufficient amount of natural light outside the room, then the lights must turn off and the blinds must open to make maximum utilization of natural light.
- If there is no sufficient light inside the room, then the lights must turn on, especially when it is dark outside the room.
- By the use of USART the system should be connected to PC for the user to manually control light in the house.

5.2 Program flowchart

The system has two flowcharts, Flowchart A and Flowchart B. Whereby Flowchart A in Figure 5.1 represents operation of lighting in the following rooms, living room, bed room, dinning room, sitting room, kitchen, bathroom and toilet.

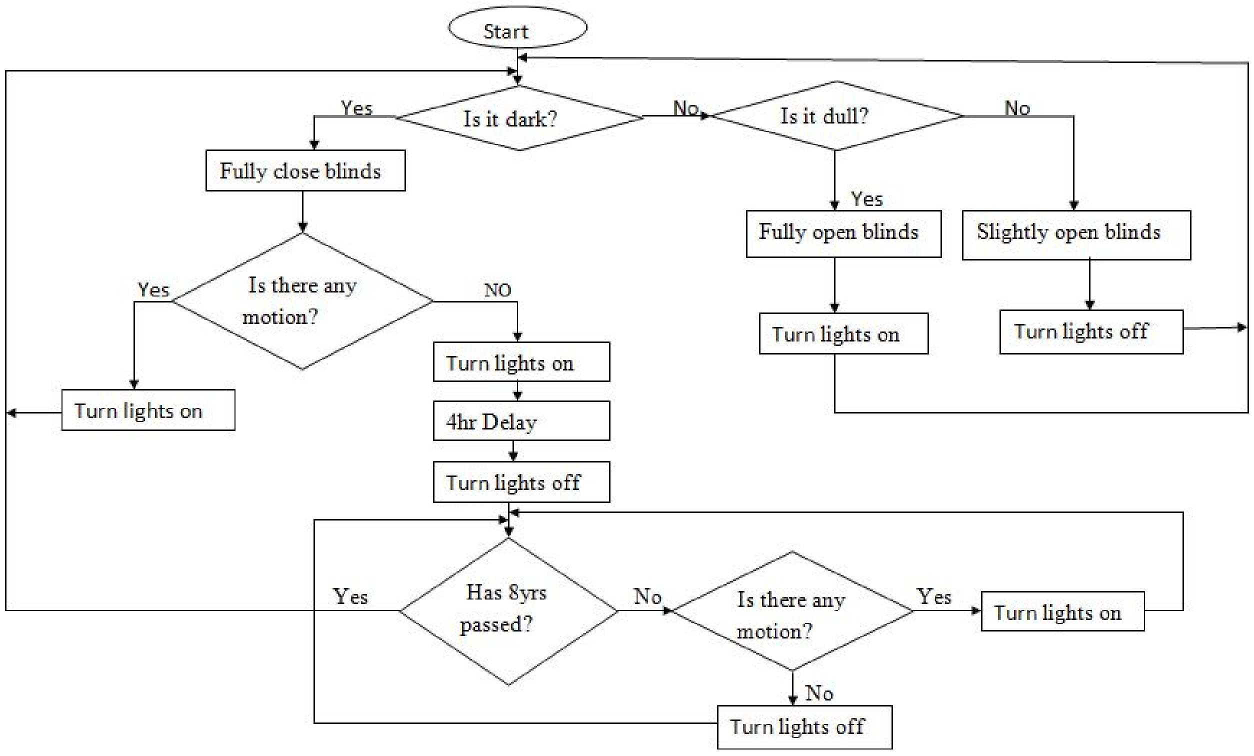


Figure 5.1: Flowchart A

Flowchart B represent corridor lights.

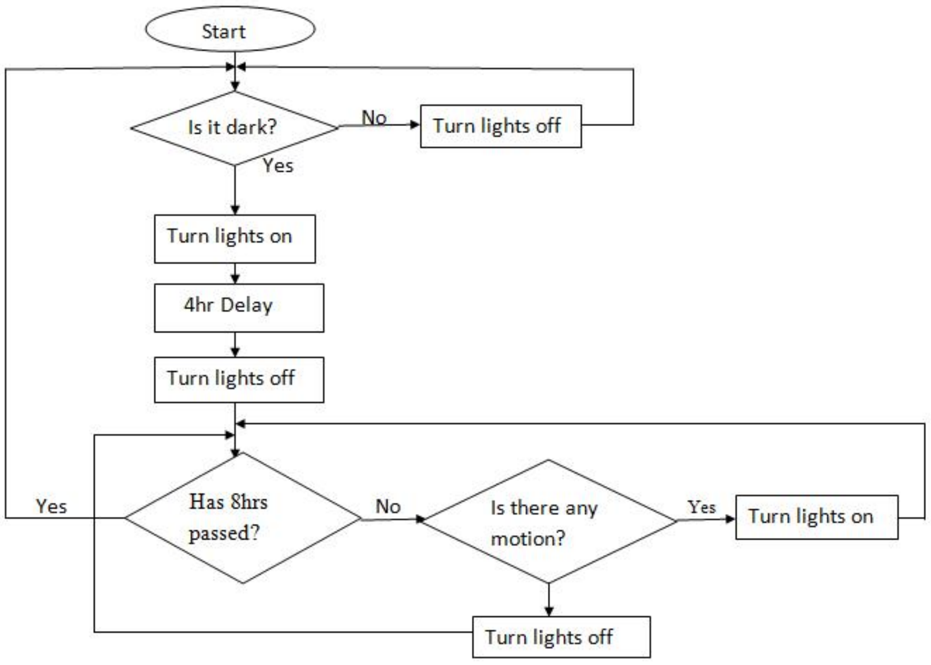


Figure 5.2: Flowchart B

5.3 How do the system work

The light detector in Figure 5.3, uses LDR sensor to sense how much light is available inside the room. The information from the LDR sensor in a form of analog signal is then taken into the microcontroller whereby the microcontroller converts the analog signal to digital signal and then interpret the digital signal. If there is no light in the room, then microcontroller send signal to the USM through motor driver to close the blinds and also sends signal to the light bulb, which turns the light bulb on (assuming it is at night).

Microcontroller also checks if the room is occupied or not with the help of motion detector, whereby motion detector uses IPD sensor to detect if

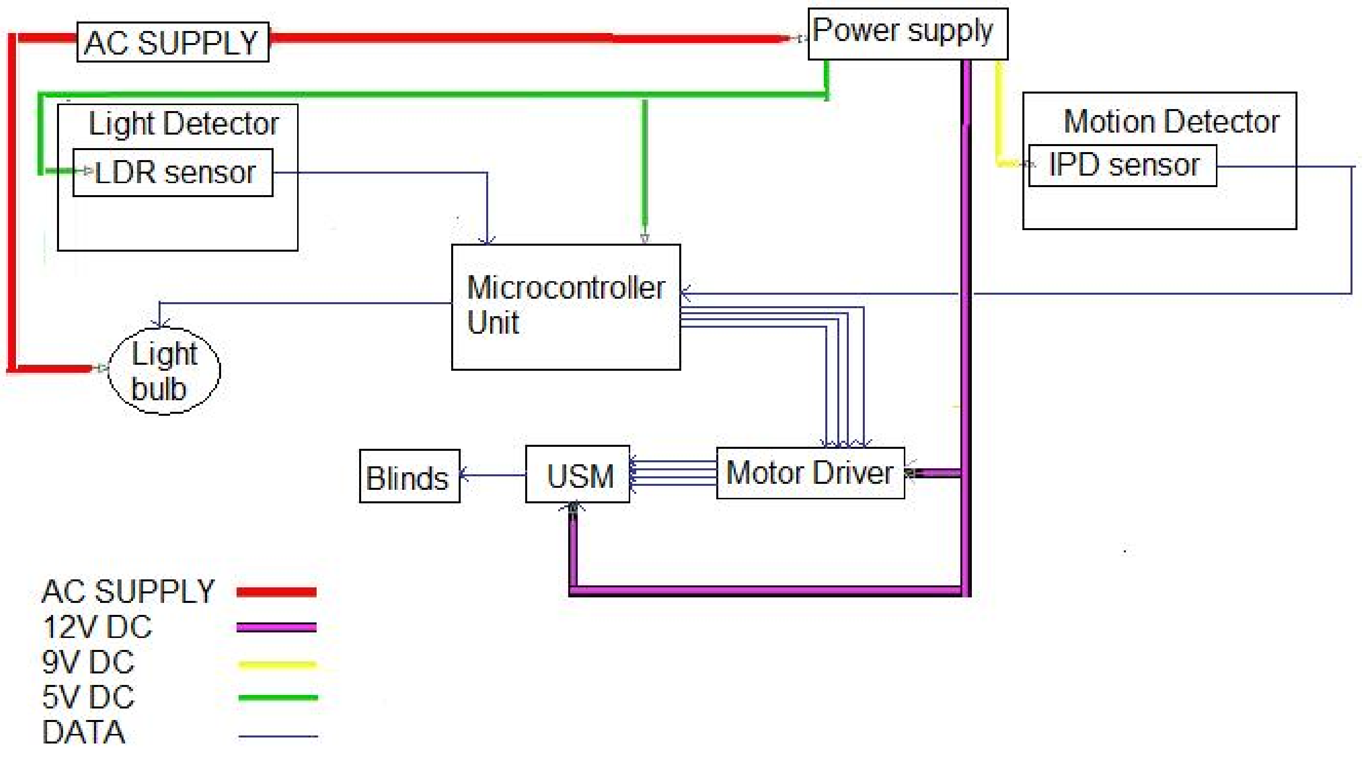


Figure 5.3: Block diagram of a typical room

there is any motion inside the room. If the room is occupied, then the light bulb turns on. Microcontroller counts the number of hours that the light bulb is turned on. After 4 hours, microcontroller turns the light bulb off and by this time it keeps on checking if there is any motion inside the room or whether the room is occupied until 8 hours has elapsed. If there is any motion detected within this 8 hours, then the light bulb will turn on and it will immediately turn off when the motion fade. But if there is no motion detected, the light bulb will remain off.

When light is available inside the room, then microcontroller send signal to USM to open blinds one step so as to let maximum utilization of natural light inside the room and it send another signal to light bulb to turn it off.

Microcontroller measures the amount of light that is available inside the room with the help of LDR sensor. If there is insufficient amount of light available inside the room (if it is dull), then microcontroller turns USM 90° to fully open the blinds and turns light bulb on. But, if there is sufficient amount of light inside the room (if it is not dull), microcontroller turns USM 45° to slightly open the blinds and turns light bulb off.

The idea of 4 hours delay and 8 hours delay that are shown in Figure 5.1 were introduced due to the fact that, if it is dark inside the room, it means it is at night and the room's light must be on for 4 hours whether there is any motion detected or not. After 4 hours, the room's light must turn off and be on only if there is a person entering the room. This was done for security purpose, so that the owner of the house can leave his or her house for days without worrying about who will put the lights on at evening and off at midnight.

Assuming it starts to be dark at 6pm, then the room's light would be on and at 10pm the room's light would turn off and by this time, for a period of 8 hours the room's light would be on only if the room is occupied and off if it is unoccupied. After 8 hours has elapsed the time would be 6am and there would be enough daylight coming into the room. Hence, system would have to loop back to the starting point.

Figure 5.2 shows the operation of corridor lights, which turns on when it is dark regardless of any motion and turns off after a delay of 4 hours. After a delay of 4 hours, the corridor lights turns on only if there is a person moving on the corridor and turns off if there is no person moving on the corridor. This last for a period of 8 hours, from there the system loops back to the start.

5.4 Hardware of the system

5.4.1 Microcontroller

Pins used

By referring to Figure 4.3 for pin configuration and Figure 5.10 for wire connection. Pin 1, Vpp is connected to 5V. Pin 33 to 36 (RB0 to RB3 respectively) are connected to the input pins of ULN2803AN motor driver to drive USM because $20mA$ sourced by each output pin of a microcontroller is very small to drive USM. Pin 2, AN0 is connected to the output of LDR in a potential divider arrangement between 5V and ground to allow initial conversion of Analog to Digital Converter (ADC). Pin 3, AN1 is connected to the output of IPD circuit which detects motion.

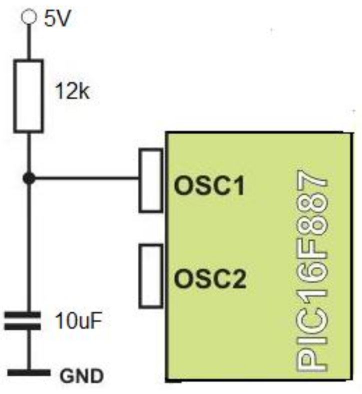


Figure 5.4: connection to OSC1

Pin 40, RB7 is connected to LED. Low current devices like LED can be interfaced directly to the output pin of microcontroller and a 330Ω is required to discard current. Pin 13, OSC1 is connected in a manner shown in Figure 5.4 to provide with extremely precise operating frequency of microcontroller.

5.4.2 Infrared Photo Detector circuit diagram

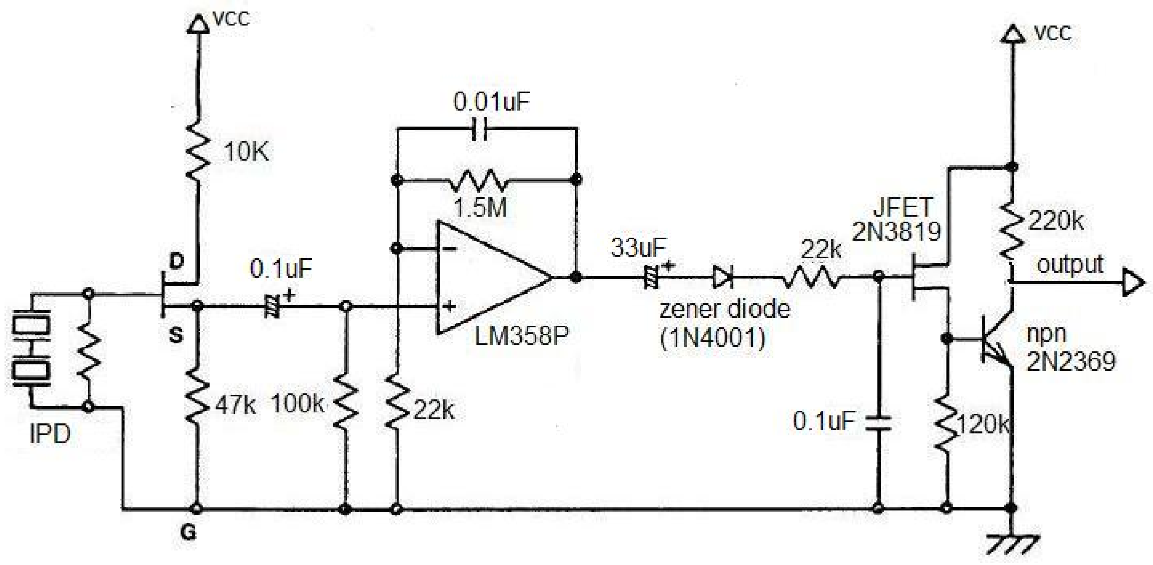


Figure 5.5: IPD circuit diagram

Figure 5.5 shows IPD circuit diagram that was constructed in order to measure output voltage of IPD. Output power signal of IPD is very small and it can not be detected by the microcontroller. Therefore, source terminal of IPD is connected to an amplifier circuit in order to increase signal power of IPD and its amplitude so that it can not be distorted by noise. Operational amplifier, LM358P is used to increase output voltage and strength of the signal while N-Channel JFET (2N3819) is used as a pre-amplifier.

Bipolar NPN transistor (2N2369) is used as an amplifier, whereby N refers to the collector, P refers to the base and the other N refers to the emitter. The collector is the larger electricity supply, the base is the gate controller and the emitter is the outlet for electricity supply. Current flowing through the gate from the collector is regulated when varying levels of current is applied from the base.

Zener diode (1N4001) is used to control the direction of flow of current from the operational amplifier, whereby it uses the resistor to ensure that the current passing through it is at least 5mA. The aim of zener diode is to regulate voltage in the circuit because it permits current to flow in the forward direction, but also allow it to flow in the reverse direction when voltage is above the required value.

5.4.3 Light detector circuit

The aim of Light detector circuit is to detect when it is dark and when it is bright. During day light when light level is high, resistance of LDR falls, allowing current to pass through it. At night, resistance of LDR becomes high and it prevents current to flow through it. Figure 5.6 shows Light detector circuit diagram. When it is dark, output voltage of LDR is less than 2.5V and greater when it is bright. Microcontroller convert analog signal from LDR to digital signal and read the output voltage of LDR. If output voltage is less than 2.5V, then LED is turned on and off if output voltage is greater than 2.5V.

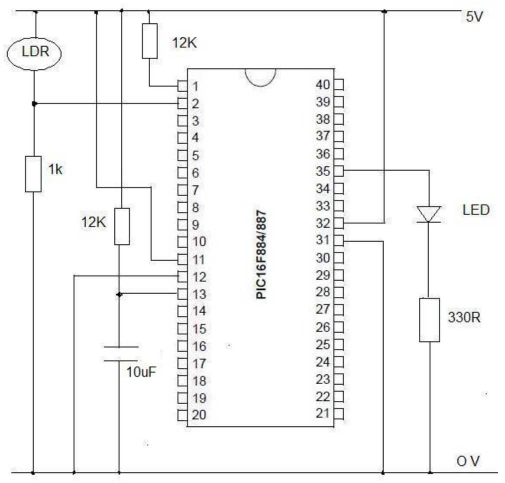


Figure 5.6: Light detector circuit diagram

5.4.4 Unipolar Stepper Motor

USM takes information from microcontroller and it uses this information to control the opening and closing of blinds. In this case, USM only rotate two steps.

- 45°, if average light is required in the room.
- 90°, if all light intensity is needed.

There is no need for the motor to rotate more than 90° because the type of blind that would be used, is the vertical window blind shown in Figure 5.7.

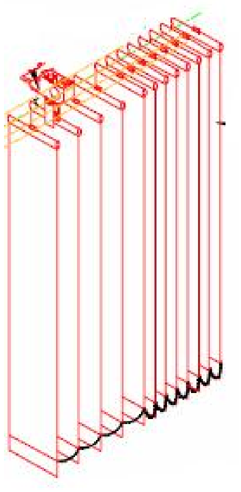


Figure 5.7: Vertical window blind

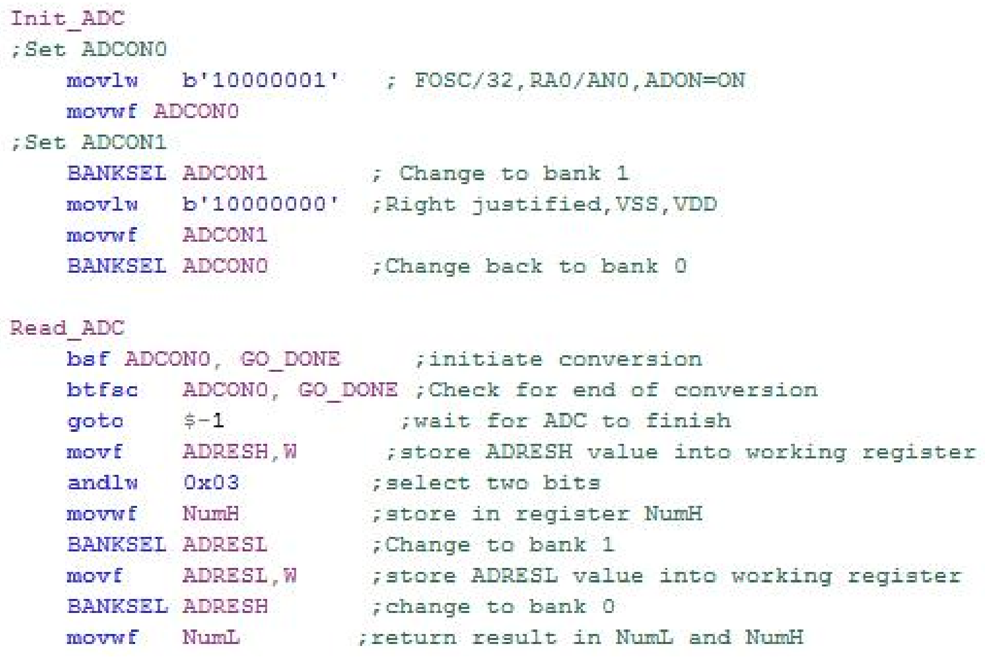
5.5 System software

5.5.1 Analog to Digital converter

ADC is used to convert analog signal into digital signal. Since the output from the two sensors is in analog form. There has to be a conversion from analog to digital because microcontrollers works with digital signals or binary bits. The fact that ADC has 10 bit resolution, meaning the output of the ADC can vary from 0 (all bits '0') to 1023 (all bits '1'). Therefore, 0V corresponds to binary 0 and 5V correspond to binary 1023. I used this mechanism to calculate binary output voltage of the two sensors.

$$V_{out}(binary) = \frac{1023V_{out}(analog)}{5} \tag{5.1}$$

The two subroutines involved when converting analog signal to digital signal are shown in Figure 5.8. The first subroutine (Init-ADC) sets the values of ADCON0 and ADCON1. The command BANKSEL is used every time



```
Init_ADC
;Set ADCON0
    movlw   b'10000001'   ; FOSC/32,RA0/AN0,ADON=ON
    movwf ADCON0
;Set ADCON1
    BANKSEL ADCON1       ; Change to bank 1
    movlw   b'10000000'  ;Right justified,VSS,VDD
    movwf   ADCON1
    BANKSEL ADCON0       ;Change back to bank 0

Read_ADC
    bsf ADCON0, GO_DONE     ;initiate conversion
    btfsc   ADCON0, GO_DONE ;Check for end of conversion
    goto    $-1              ;wait for ADC to finish
    movf    ADRESH,W          ;store ADRESH value into working register
    andlw   0x03         ;select two bits
    movwf   NumH         ;store in register NumH
    BANKSEL ADRESL       ;Change to bank 1
    movf    ADRESL,W     ;store ADRESL value into working register
    BANKSEL ADRESH       ;change to bank 0
    movwf   NumL        ;return result in NumL and NumH
```

Figure 5.8: Subroutines for ADC

before data in the working register could be forwarded to ADCON1 because the two registers (ADC0N0 and ADC0N1) are in different registers bank, hence correct bank has to be chosen before register is accessed and then return to the original register bank.

The second subroutine (Read-ADC) reads the analog input. The first instruction initiates conversion while the second and third instruction waits for conversion to be completed. Instructions 4, 5 and 6 is where converted value is read and stored for later use. ANDing the value in ADRESH with 0x03 allows reading of only two bits while the other 8 bits are read in ADRESL. Again there is bank switch because ADRESL and ADRESH are in different register bank. The final converted value is stored in registers NumL and NumH. This is because registers store only 8 bits and 10 bits require two registers.

5.6 Serial Communication

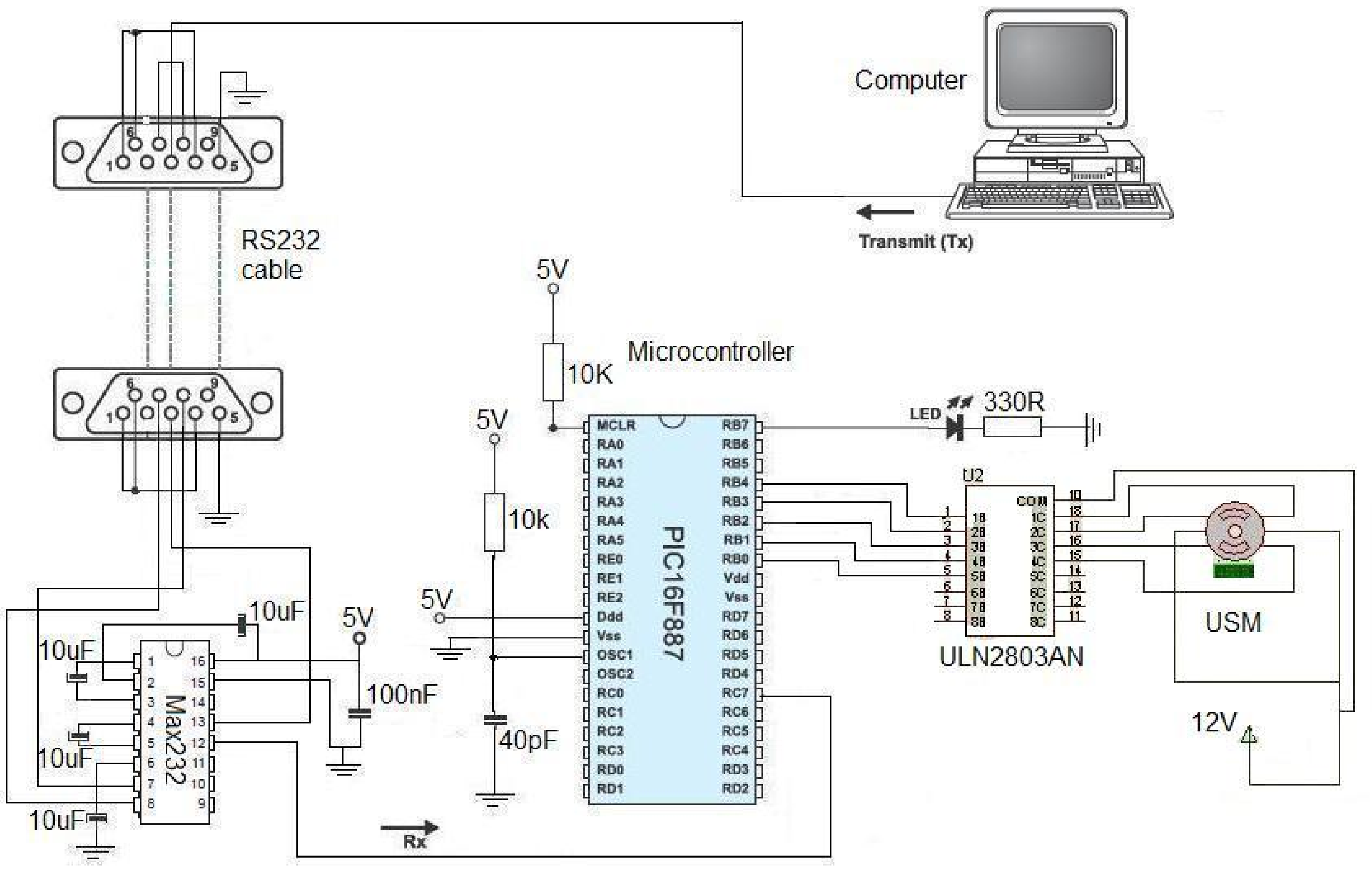


Figure 5.9: Serial communication

Referring to Figure 5.9 for serial communication, data from computer is transmitted through RS232 cable using wire Transmit(TX) and it splits to three different pins of 9 pin D type female connector, which are TxD, RTS and CTS. TxD transmits data to DCE, RTS ask for a request to send outgoing flow control signal and CTS checks if it is clear to send incoming flow control signal.

TxD, RTS and CTS of 9 pin D type male connector are connected to R1IN (Pin 13), R2IN (Pin 8) and T2OUT (Pin 7) of MAX232 respectively. This is because serial RS232 communication works with voltages which are not compatible with computer logic voltages, hence MAX232 is used to convert signal from RS232 to signal suitable for use in Transistor Transistor Logic (TTL) compatible digital logic circuits.

R1IN and R2IN receives input from RS232 while T2OUT drives the data to the output. Pin 12, R1OUT of MAX232 is connected to pin 26, RX/RC7 of PIC16F887. They are connected by wire Rx. R1OUT output the received data from RS232 while RX/RC7 receives data from MAX232. RB0, RB1, RB2 and RB3 are connected to 1B, 2B, 3B and 4B of motor driver respectively, to drive USM.

5.6.1 Transmitting Data

For transmission of data, java language was used. There are two tasks that need to be accomplished in order to perfect transmission of data. First, the number of ports in a computer must be identified using static factory method $CommPortIdentifier.getPortIdentifiers()$ to return the list of available ports. The complete code is shown in Appendix E. The second task is to set a serial port and open it for input or output stream. There are four steps that must be performed in the second task and they are:

- Receiving CommPortIdentifier from the system with specific name because received identifier may be in use by another application.

- Opening CommPort if received CommPortIdentifier is not in use by different application.
- Setting the serial communications parameters, such as baud rate, number of data bits, number of stop bits and parity bit.
- Retrieving OutputStream for sending raw bytes by calling CommPortSender class shown in Appendix F.

The code for second task is shown in Appendix G.

5.6.2 Receiving Data

For receiving data, Assembly language was used. Again, there are two tasks that must be done and those task are; setting up USART connection and setting asynchronous reception. In USART connection the following steps were made:

- Initializing SPBRGH, SPBRG register pair and the BRGH and BRG16 bits to achieve the desired baud rate.
- Setting SPEN bit of RCSTA to enable the synchronous master serial port.
- Setting CREN bit of RCSTS to enable receive mode.

The following steps were made to set asynchronous reception:

- Setting TRISC$< 7 >$ in order to configure RC7/RX/DT pin as input.

- Enabling asynchronous serial port by clearing bit SYNC and setting bit SPEN.
- Checking whether bit RCIF is set when reception is completed.
- Reading the 8-bit received data by reading the RCREG register.

The complete code for receiving data and performing required instructions (opening or closing blinds and turning on or off the lights) is shown in Appendix H. Photographs of complete circuit of serial communication are shown on Appendix I.

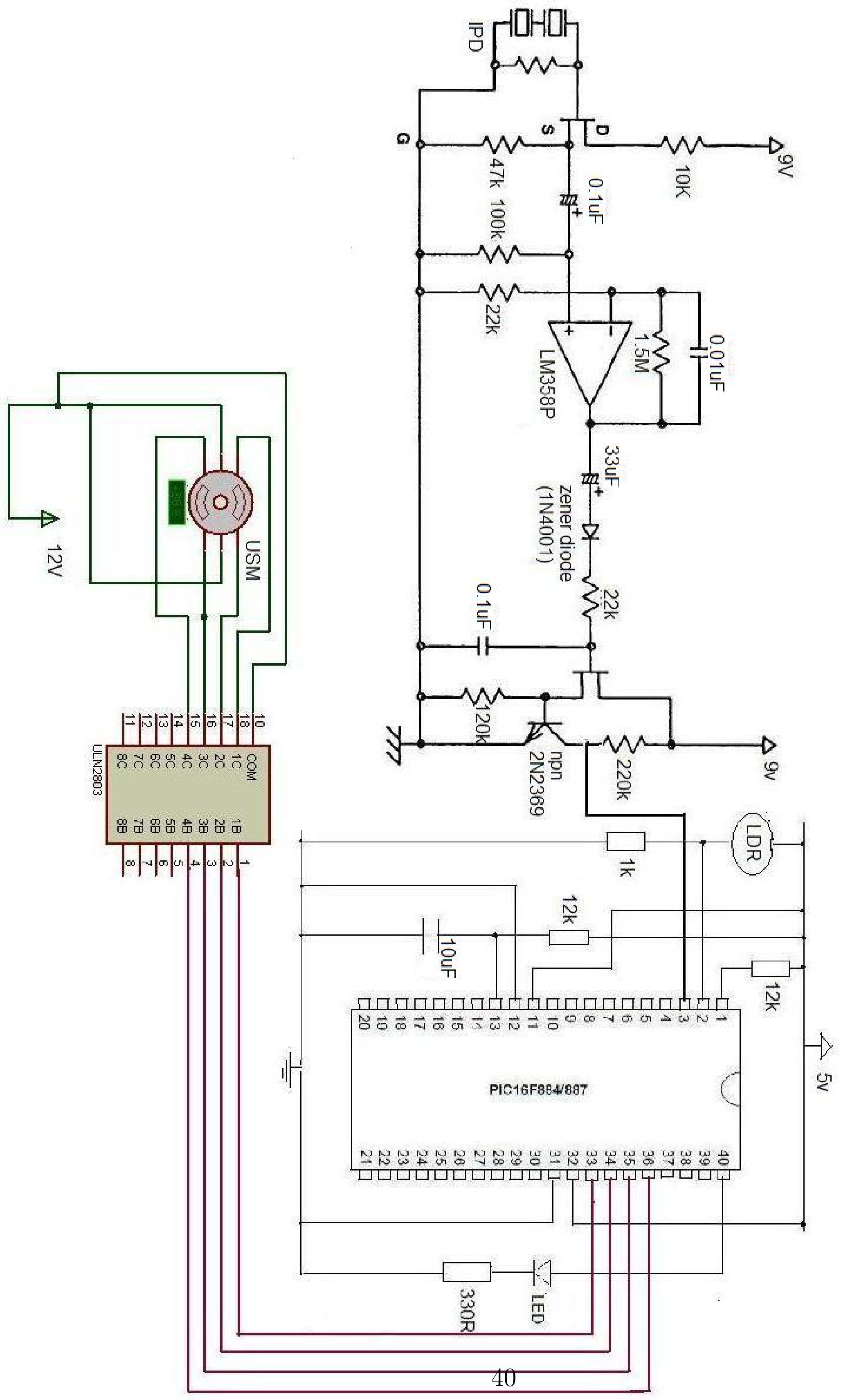


Figure 5.10: Complete circuit diagram of a typical room

Chapter 6

Results

6.1 Testing and measurement of equipment

The devises to be used were tested to see if they are in good condition so that there would be no error on the final system after they have been assembled together.

6.2 Light Dependent Resistor

Output voltage of LDR in light and in dark shade was measured using different resistors. The aim was to find out how much voltage it output at day light and during the night. I constructed voltage divider as shown in Figure 6.1, whereby the output voltage is given by:

$$V_{out} = \frac{R_{LDR}}{R_{LDR+R}} V_{in} \tag{6.1}$$

The fact that there is a wide range of output voltage between V_{out} in dark and V_{out} in light when using the resistance of 1KΩ (see Table 6.1). I decided

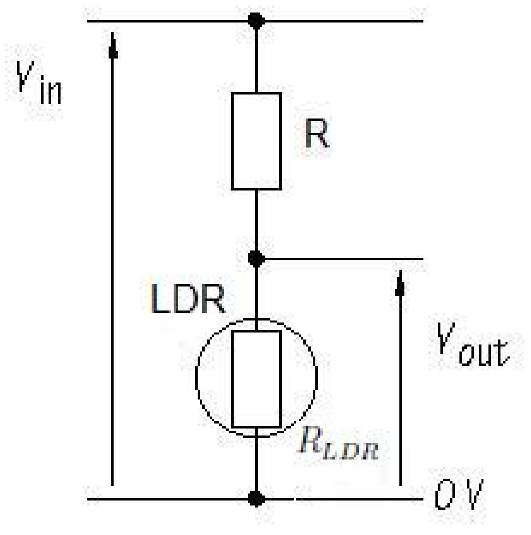


Figure 6.1: Voltage divider circuit

to use 1KΩ in a potential divider arrangement with LDR, in light detector circuit because there would be a small distortion when the sensor is detecting whether it is bright or dark compared to other resistors.

Fixed Resistor value	V_{out} in dark shade (V)	V_{out} in light (V)
100Ω	2.433	4.80 to 5
1KΩ	0.089	4.80 to 5
10KΩ	0.105	4.5
100KΩ	0.032	2.51 to 3.2

Table 6.1: Values of output voltage obtained in a voltage divider circuit.

6.3 Infrared Photo Detector

IPD circuit shown in Figure 5.5 was constructed and output voltage of IPD was measured when there is motion, using different supply voltages because supply voltage for IPD can vary from 2.0V to 15.0V. Results obtained are shown on the graph of Output voltage vs Supply voltage in Figure 6.2. When Supply voltage increases, Output voltage also increases until it reaches saturation at 4mV when Supply voltage is 9V. Hence, 9V power supply was used in IPD circuit.

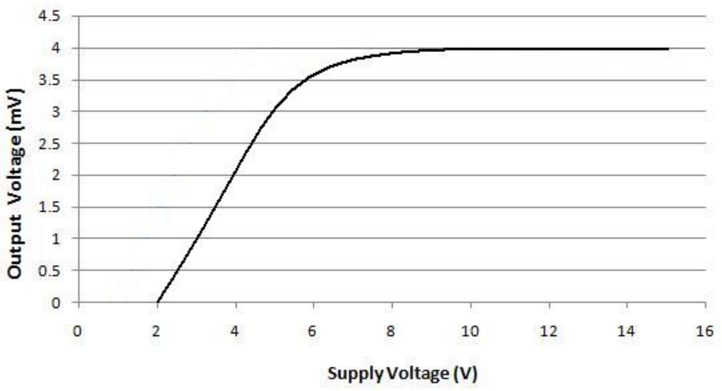


Figure 6.2: Output voltage vs Supply voltage.

6.4 Unipolar Stepper Motor

Like I mentioned earlier in Chapter 4 that for USM to rotate, different pulse sequence of ones and zeros have to be applied to it. Different pulse sequence were applied to the USM, to determine how much in degrees it would turn. Table 6.2 shows how much in degrees USM turn when certain pulse sequence is applied to it and in which direction it turns.

Applied sequence	Angle of turning	Direction
0000	$0°$	No direction
0001	$+45°$	Clockwise
0010	$-45°$	Anticlockwise
0011	No rotation	
0100	$-135°$	Anticlockwise
0101	No rotation	
0110	$-90°$	Anticlockwise
0111	$-45°$	Anticlockwise
1000	$+135°$	Clockwise
1001	$+90°$	Clockwise
1010	No rotation	
1011	$+45°$	Clockwise
1100	$+180°$	Clockwise
1101	$+135°$	Clockwise
1110	$+90°$	Anticlockwise
1001	No rotation	

Table 6.2: Pulse sequence applied to a USM.

6.5 Simulation

6.5.1 Turning Unipolar Stepper Motor

The code for turning USM 90° was loaded into MPLAB IDE v8.30[1] software and simulated. From the simulation it was observed that the code is correct. The code is shown in Appendix A. Circuit diagram for turning USM 90° was constructed and tested on TST-ISIS7 Professional[2] software using

[1]Windows Operating System software program that runs on a PC to develop applications for Microchip microcontrollers and digital signal controllers.

[2]software for automated design of electronic circuits.

PIC16F877A because PIC16F887 is not available on pick list of the software. Therefore PIC16F877A was used because it is much similar to PIC16F887.

Result obtained in testing circuit diagram for turning USM 90° on TST-ISIS7 Professional software is shown in Figure 6.3. Pins with red dot shows that there is current flowing through them while pins with blue dot shows that there is no current flowing through them. A red dot represent a 1, which turns on the current through a motor winding while a blue dot represent a 0, which terminate current from flowing through a motor winding. Therefore, it is easily seen that pulse sequence applied is 1001. +90.0 on the USM shows that USM has rotated 90°.

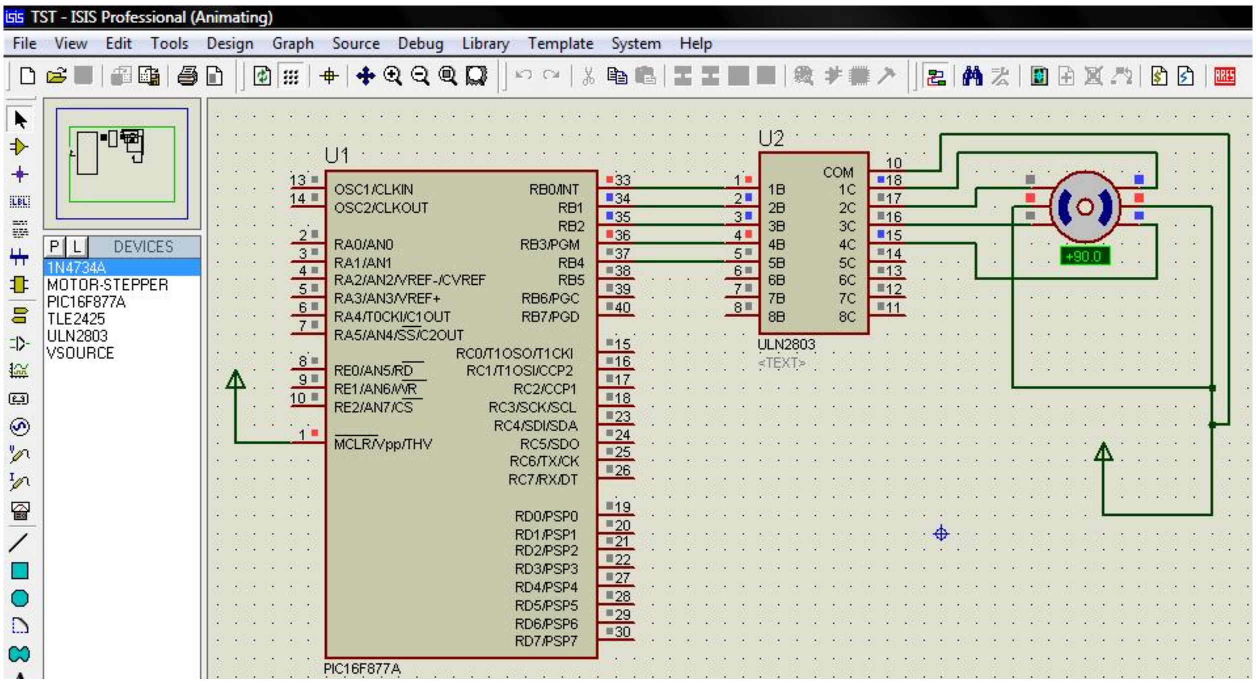
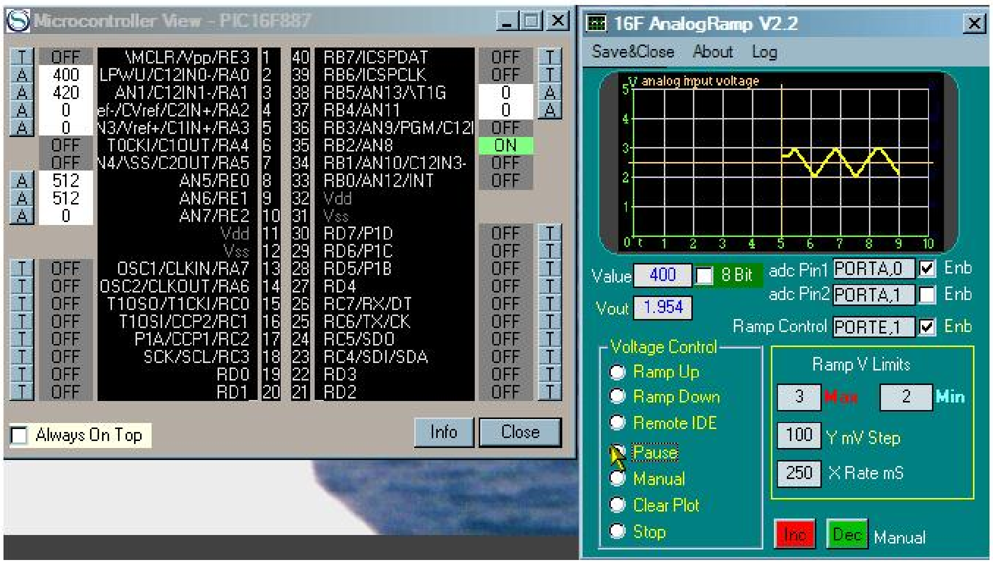

TST - ISIS Professional (Animating)
File View Edit Tools Design Graph Source Debug Library Template System Help
DEVICES
1N4734A
MOTOR-STEPPER
PIC16F877A
TLE2425
ULN2803
VSOURCE
U1
PIC16F877A
U2
ULN2803
<TEXT>
+90.0

Figure 6.3: Circuit diagram for turning USM

6.5.2 Light detector

The code for light detector was loaded into MPLAB IDE v8.30 software and simulated. Moreover, the code was simulated on Oshonsoft simulator[3] before assembling of the whole system. From Oshonsoft simulator shown in Figure 6.4, analog output voltage of LDR entering at AD channel 0 input (ANO, Pin 2 of PIC16F887) is 1.954V which its corresponding binary digital output voltage is 400, Therefore, it is is easily seen that when the output voltage of LDR is less than 2.5V, then LED (at RB2, Pin 35 of PIC16F887) is turned on.


Microcontroller View - PIC16F887

T	OFF	\MCLR/Vpp/RE3	1	40	RB7/ICSPDAT	OFF	T
A	400	LPWU/C12IN0-/RA0	2	39	RB6/ICSPCLK	OFF	T
A	420	AN1/C12IN1-/RA1	3	38	RB5/AN13/\T1G	0	A
A	0	ef-/CVref/C2IN+/RA2	4	37	RB4/AN11	0	A
A	0	N3/Vref+/C1IN+/RA3	5	36	RB3/AN9/PGM/C12I	OFF	
	OFF	T0CKI/C1OUT/RA4	6	35	RB2/AN8	ON	
	OFF	N4/\SS/C2OUT/RA5	7	34	RB1/AN10/C12IN3-	OFF	
A	512	AN5/RE0	8	33	RB0/AN12/INT	OFF	
A	512	AN6/RE1	9	32	Vdd		
A	0	AN7/RE2	10	31	Vss		
		Vdd	11	30	RD7/P1D	OFF	T
		Vss	12	29	RD6/P1C	OFF	T
T	OFF	OSC1/CLKIN/RA7	13	28	RD5/P1B	OFF	T
T	OFF	OSC2/CLKOUT/RA6	14	27	RD4	OFF	T
T	OFF	T1OSO/T1CKI/RC0	15	26	RC7/RX/DT	OFF	T
T	OFF	T1OSI/CCP2/RC1	16	25	RC6/TX/CK	OFF	T
T	OFF	P1A/CCP1/RC2	17	24	RC5/SDO	OFF	T
T	OFF	SCK/SCL/RC3	18	23	RC4/SDI/SDA	OFF	T
T	OFF	RD0	19	22	RD3	OFF	T
T	OFF	RD1	20	21	RD2	OFF	T

☐ Always On Top

16F AnalogRamp V2.2

Value 400 ☐ 8 Bit
Vout 1.954
adc Pin1 PORTA,0 ☑ Enb
adc Pin2 PORTA,1 ☐ Enb
Ramp Control PORTE,1 ☑ Enb

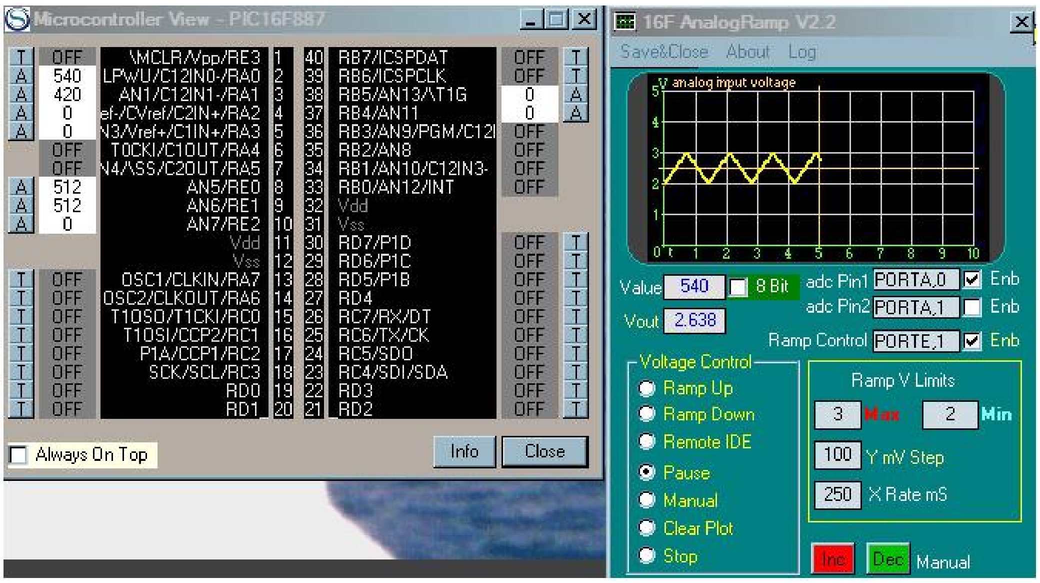
Ramp V Limits
3 Max 2 Min
100 Y mV Step
250 X Rate mS

Figure 6.4: Result of Oshonsoft simulator

Figure 6.5, shows that when the output voltage of LDR entering at AD channel 0 input is greater than 2.5V, then LED (at RB2, Pin 35 of PIC16F887) is turned off. In this case, analog output voltage of LDR is 2.638V which

[3]PIC Simulator IDE that supplies PIC developers with user-friendly graphical development environment for Windows with integrated simulator, basic compiler, assembler, disassembler and debugger.

its corresponding binary digital output voltage is 540. After simulation, the code was loaded into the microcontroller and Light detector circuit was constructed. Light detector circuit in dark shade and in daylight can be seen in Appendix C .


Microcontroller View - PIC16F887

T	OFF	\MCLR/Vpp/RE3	1	40	RB7/ICSPDAT	OFF	T
A	540	LPWU/C12IN0-/RA0	2	39	RB6/ICSPCLK	OFF	T
A	420	AN1/C12IN1-/RA1	3	38	RB5/AN13/\T1G	0	A
A	0	ef-/CVref/C2IN+/RA2	4	37	RB4/AN11	0	A
A	0	N3/Vref+/C1IN+/RA3	5	36	RB3/AN9/PGM/C12I	OFF	
	OFF	T0CKI/C1OUT/RA4	6	35	RB2/AN8	OFF	
	OFF	N4/\SS/C2OUT/RA5	7	34	RB1/AN10/C12IN3-	OFF	
A	512	AN5/RE0	8	33	RB0/AN12/INT	OFF	
A	512	AN6/RE1	9	32	Vdd		
A	0	AN7/RE2	10	31	Vss		
		Vdd	11	30	RD7/P1D	OFF	T
		Vss	12	29	RD6/P1C	OFF	T
T	OFF	OSC1/CLKIN/RA7	13	28	RD5/P1B	OFF	T
T	OFF	OSC2/CLKOUT/RA6	14	27	RD4	OFF	T
T	OFF	T1OSO/T1CKI/RC0	15	26	RC7/RX/DT	OFF	T
T	OFF	T1OSI/CCP2/RC1	16	25	RC6/TX/CK	OFF	T
T	OFF	P1A/CCP1/RC2	17	24	RC5/SDO	OFF	T
T	OFF	SCK/SCL/RC3	18	23	RC4/SDI/SDA	OFF	T
T	OFF	RD0	19	22	RD3	OFF	T
T	OFF	RD1	20	21	RD2	OFF	T

☐ Always On Top

16F AnalogRamp V2.2

Value 540 ☐ 8 Bit
Vout 2.638
adc Pin1 PORTA,0 ✓ Enb
adc Pin2 PORTA,1 ☐ Enb
Ramp Control PORTE,1 ✓ Enb

Voltage Control: ○ Ramp Up ○ Ramp Down ○ Remote IDE ● Pause ○ Manual ○ Clear Plot ○ Stop

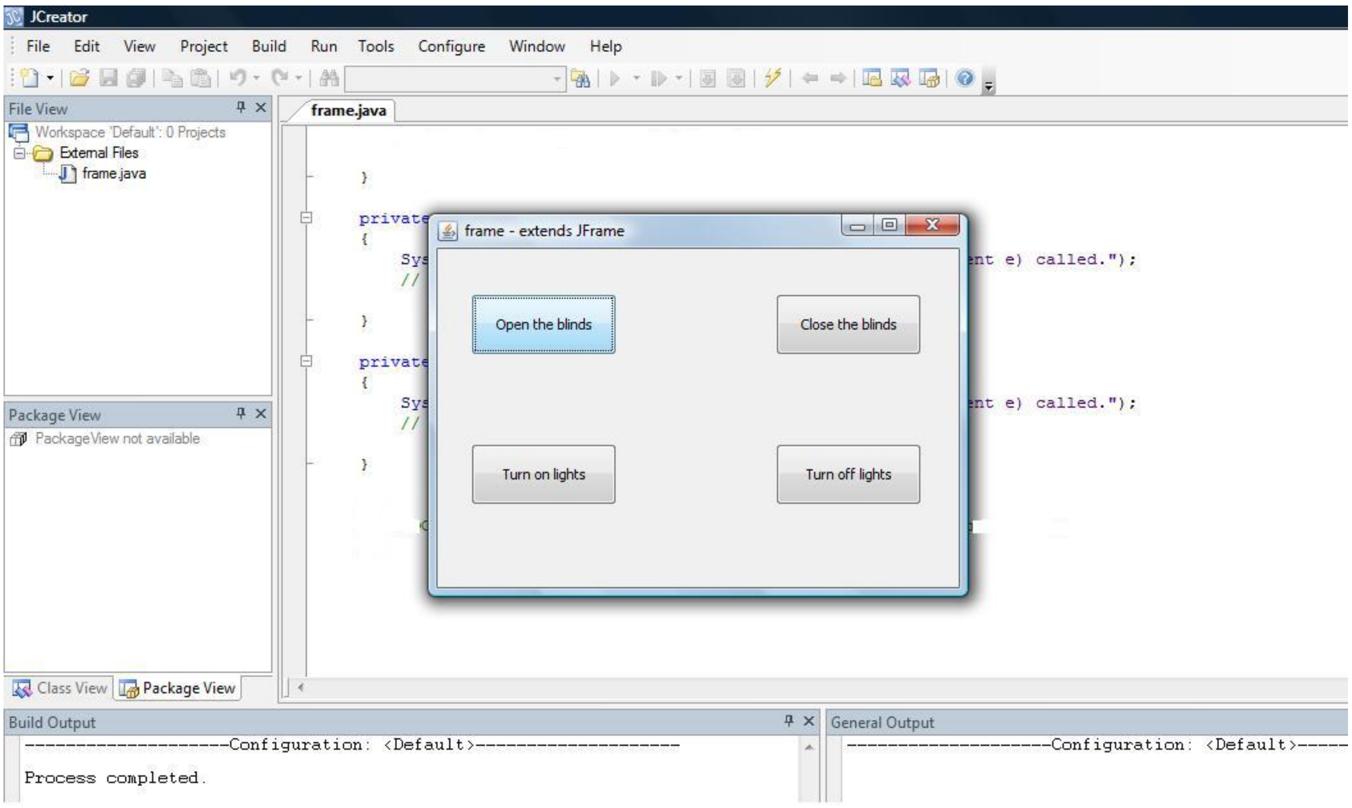
Ramp V Limits: 3 Max 2 Min
100 Y mV Step
250 X Rate mS

Figure 6.5: Result of Oshonsoft simulator

6.6 Serial Communication

6.6.1 Computer Interface

With the help of USART, the system is connected to laptops and PCs for the user to manually control light in the house. Figure 6.6 shows the interface that is created to appear on the computer screen. The interface is user friendly, the user can open or close the blinds and also turn on or off the lights by just clicking on the correct button. The interface was created



```
    }

    private
    {
        Sys                                        nt e) called.");
        //

    }

    private
    {
        Sys                                        nt e) called.");
        //

    }
```

```
--------------------Configuration: <Default>--------------------

Process completed.
```

```
--------------------Configuration: <Default>-----
```

Figure 6.6: Computer interface

using JFrameBuilder[4] software and it was compiled using JCreator 4.50 Pro software[5]. The code for creating computer interface is shown in Appendix D.

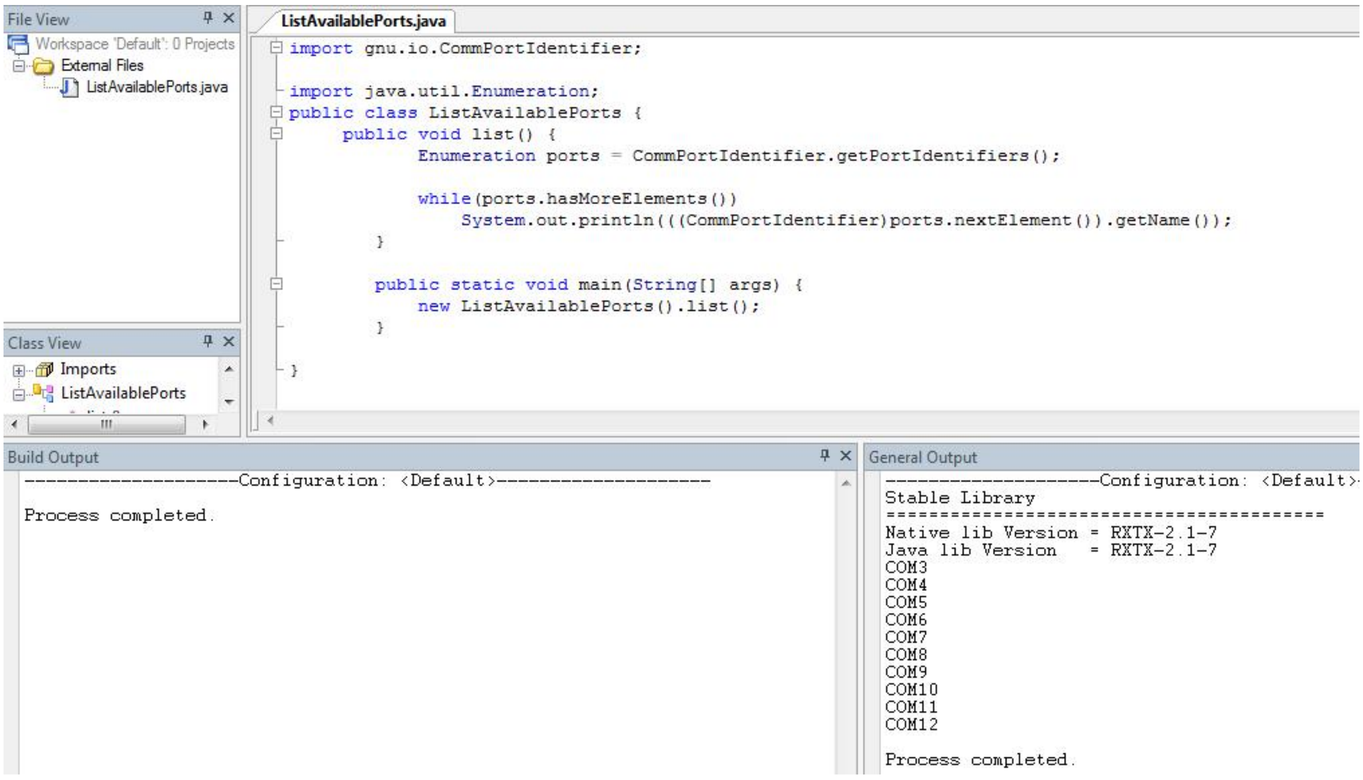
6.6.2 Available Ports

In order to run serial communication, it is important to know the number of available ports, so that it can be easily seen whether there is any free ports

[4]An easy-to-use visual Java GUI Builder for Java Swing applications. The Java GUI designer enables Java developers to create sophisticated GUI applications using drag-and-drop interface.

[5]A powerful interactive development environment for java technologies that provides with JDK tools to compile, debug and run java projects.

that can be used by serial communication application. The code shown in Figure 6.7 was tested on JCreator 4.50 Pro software. General output in Figure 6.7 shows the number of available ports, from COM3 to COM12. Number of listed ports depends on the number of available ports in the operating system.



```
import gnu.io.CommPortIdentifier;

import java.util.Enumeration;
public class ListAvailablePorts {
        public void list() {
                Enumeration ports = CommPortIdentifier.getPortIdentifiers();

                while(ports.hasMoreElements())
                    System.out.println(((CommPortIdentifier)ports.nextElement()).getName());
            }

            public static void main(String[] args) {
                new ListAvailablePorts().list();
            }

}
```

```
--------------------Configuration: <Default>--------------------

Process completed.
```

```
--------------------Configuration: <Default>
Stable Library
=========================================
Native lib Version = RXTX-2.1-7
Java lib Version   = RXTX-2.1-7
COM3
COM4
COM5
COM6
COM7
COM8
COM9
COM10
COM11
COM12

Process completed.
```

Figure 6.7: Available Ports

Chapter 7

Conclusion

7.1 Performance of the system

Automatic light control was successfully accomplished. The light detector circuit was able to distinguish between daylight and night, by yielding a satisfactory response to varying light intensities in the house. The results can be seen in Appendix C.

Due to well organized programming and good use of memory resource the entire programm for light detector circuit used few memory space available in the microcontroller as it can be seen in Figure 5.6 that only 8 pins out of 40 pins of the microcotroller were used. This also shows that power requirement of the microcontroller was too low. As the result, the cost of ownership is less than cost of initial purchase.

For the entire system, looking at the cost analysis in Appendix J, to built the system, M651.96 was used. Therefore, it is clearly seen that the system is affordable. The light detector circuit is reliable since it is trusted to do what is expected out of it, this is because it was tested in the lab whereby temperature was varied from 16°C to 32°C and the result was consistent. Therefore, it does not depend on temperature variation, hence it can operate well in winter and summer.

The system can be easily upgraded and extended because of more available memory space on the microcontroller. Although not all of the performance requirement were met, but significant requirement stated above were successfully carried out.

7.2 Disadvantages of the system

There are three problems encountered in motion detector circuit, the motion detector circuit was not consistent with range, movement and temperature. It does not detect when a human being is two meters away, slow or slight movement is not detected. Again the results obtained in IPD circuit are not temperature independent, they vary with varying temperature.

In serial communication, whereby the system is connected to laptops or computers for the user to manually control the lights in the house, did not perform intended functions properly. After turning the light on, an attempt to turn them of fails, this might have been brought by incomplete rxtx packages and wrong specification of RS232 communication protocol between end point device and the serial communication application.

7.3 Way forward

The system is connected to laptops and computers for user to manually control lighting in the house. But for future work, it would be more useful if the system could be connected to laptops and computers for modification or reprogramming of the system to suit the needs and requirements of the user. Also to provide with presets for the user to change the illuminance and blind position according to his or her needs. A more sophisticated method is to use photodiodes to control the lighting system based on available ambient lighting and also to include dimmer circuitry in a light detector circuit in order to dim the lights instead of turning them off if insufficient amount of light is required.

The dimmer circuitry should also allow the user to decrease or increase lamp brightness according to his or her will. Since the system would be using lamps for lighting instead of LEDs which were just used for demonstration. It would be far more better if the system could use two types of lamps Compact Fluorescent Lamps (CFL) and Incandescent Lamps (GLS), and it must depend on the user which type should be on or off during system operation. Since the light detector circuit used few memory space of microcontroller, it would be wise to find ways of implementing the system using small microcontrollers, so as to reduce cost of big microcontrollers.

In serial communication it would be wise if the system could be designed in such away that the user get a feedback from the microcontroller that appears on the computer screen. It would be helpful because the user can know when there is a problem with the system.

Appendix A

Appendix: Code for turning motor 90° clockwise

```
;PROGRAM FUNCTION:turning motor
    list        P=16F877a
    include    "P16F877a.inc"
                __config 0x3D18

    CBLOCK 20h   ; Temporary storage
         dc1
     ENDC

    org     0x0000
    goto Start

;Program Start:
 Start
    call Init                ;to initialize portb
```

```
;------------------------------------------------------------------------
;Subroutines: wait
  ; Outer loop iteration count
       movlw   5            ;count 5 times
       movwf   dc1
       decfsz  dc1,F        ;Decrement 5, Skip if 0
       goto    $-1          ;wait for loop count to finish
       return
;------------------------------------------------------------------------
Init
       clrw                 ; Zero.
       movwf   PORTB        ; Ensure PORTB is zero before we enable it.
       bsf     STATUS,RP0   ; Select Bank 1
       movlw   0xF0         ; Set port B bits 0-3 as outputs
       movwf   TRISB        ; Set TRISB register.
       bcf     STATUS,RP0   ; Select Bank 0
;------------------------------------------------------------------------
Main
  ;applying different sequence of pulses to drive motor
       movlw b'1001'
       movwf PORTB
       call wait

 end
```

Appendix B

Code for light detector

```
;PROGRAM FUNCTION:light detector
 list p=16f887 ; list directive to define processor
 #include <p16f887.inc>  ; processor specific variable definitions

__config _CONFIG1, _LVP_OFF & _FCMEN_ON & _IESO_OFF & _BOR_OFF &
_CPD_OFF & _CP_OFF & _MCLRE_ON & _PWRTE_ON & _WDT_OFF &
_INTRC_OSC_NOCLKOUT
 __config _CONFIG2, _WRT_OFF & _BOR21V



;Declarations:
   NumH    equu    20h  ;General purpose registers
   NumL    equ     21h

    org     0x0000
    goto Start
;Program Start: Start
    call Init     ;for initialization
Main
   ; Read_ADC
    bsf ADCON0, GO_DONE     ;initiate conversion
    btfsc   ADCON0, GO_DONE ;Check for end of conversion
    goto    $-1            ;wait for ADC to finish
    movf    ADRESH,W       ;store ADRESH value into working register
    andlw   0x03          ;select two bits
    movwf   NumH          ;store in register NumH
    BANKSEL ADRESL        ;Change to bank 1
    movf    ADRESL,W      ;store ADRESL value into working register
    BANKSEL ADRESH        ;change to bank 0
    movwf   NumL         ;return result in NumL and NumH

;Check if it is dark or bright
    btfss NumH,1       ;check if output voltage exit 2.5 volts
    goto led_on
    bcf PORTB,2       ; turn Off the LED
    goto Main
led_on
    bsf PORTB,2      ;turn on the LED
    goto Main

;Subroutines: Init ;Initializing porta
```

```
    BCF STATUS, RP0 ;
    BCF STATUS, RP1 ; Bank0
    CLRF PORTA      ; Initialize PORTA clearing output data latches
    BSF STATUS, RP0 ; Select Bank 1
    MOVLW 0x06      ; Configure all pinss
    MOVWF ADCON1    ; as digital inputs
    MOVLW 0xCF      ; Value used to initialize data  direction
    MOVWF TRISA     ; Set RA<3:0> as inputs RA<5:4> as outputs
                    ;TRISA<7:6>are always read as '0'

 ; Initializing portb
    clrw                    ; zerp
    movwf   PORTB           ; Ensure PORTB is zero before we enable it.
    bsf     STATUS,RP0      ; Select Bank 1
    movlw   0xF0            ; Set port B bits 0-3 as outputs
    movwf   TRISB           ; Set TRISB register.
    bcf     STATUS,RP0      ; Select Bank 0

;initializing ADC ;Set ADCON0
    movlw   b'10000001'   ; FOSC/32,RA0/AN0,ADON=ON
    movwf ADCON0
;Set ADCON1
    BANKSEL ADCON1       ; Change to bank 1
    movlw   b'10000000'  ;Right justified,VSS,VDD
    movwf   ADCON1
    BANKSEL ADCON0       ;Change back to bank 0
```

```
return


END
```

Appendix C

photographs of light detector circuit

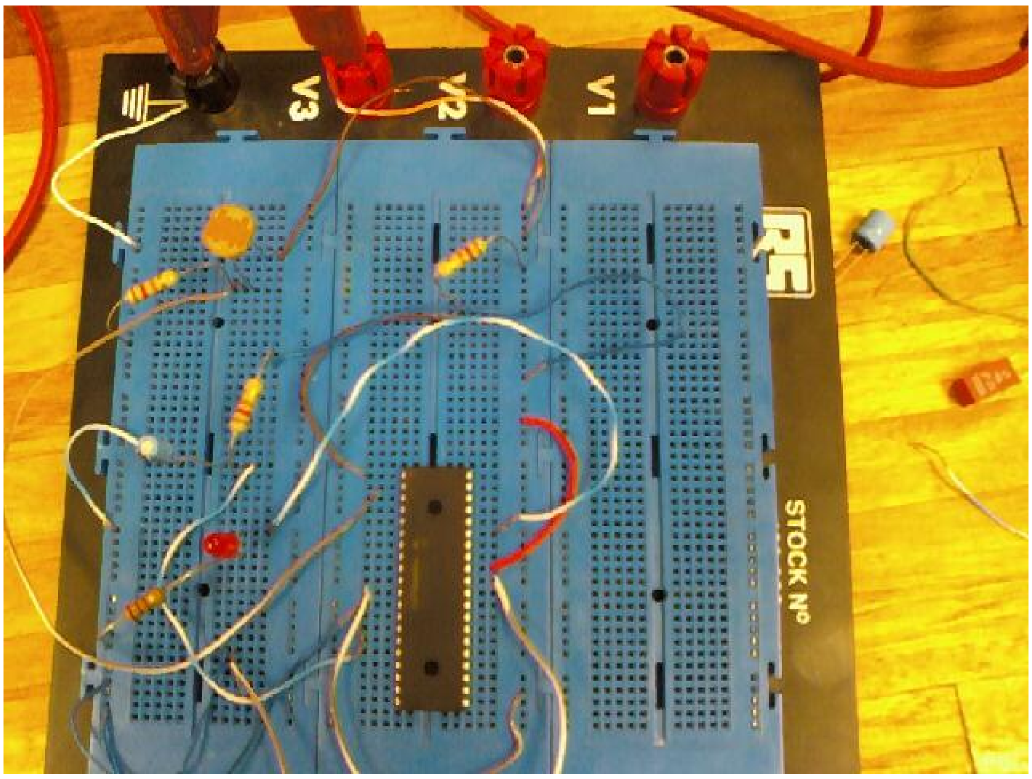


Figure C.1: light detector circuit when it is bright.

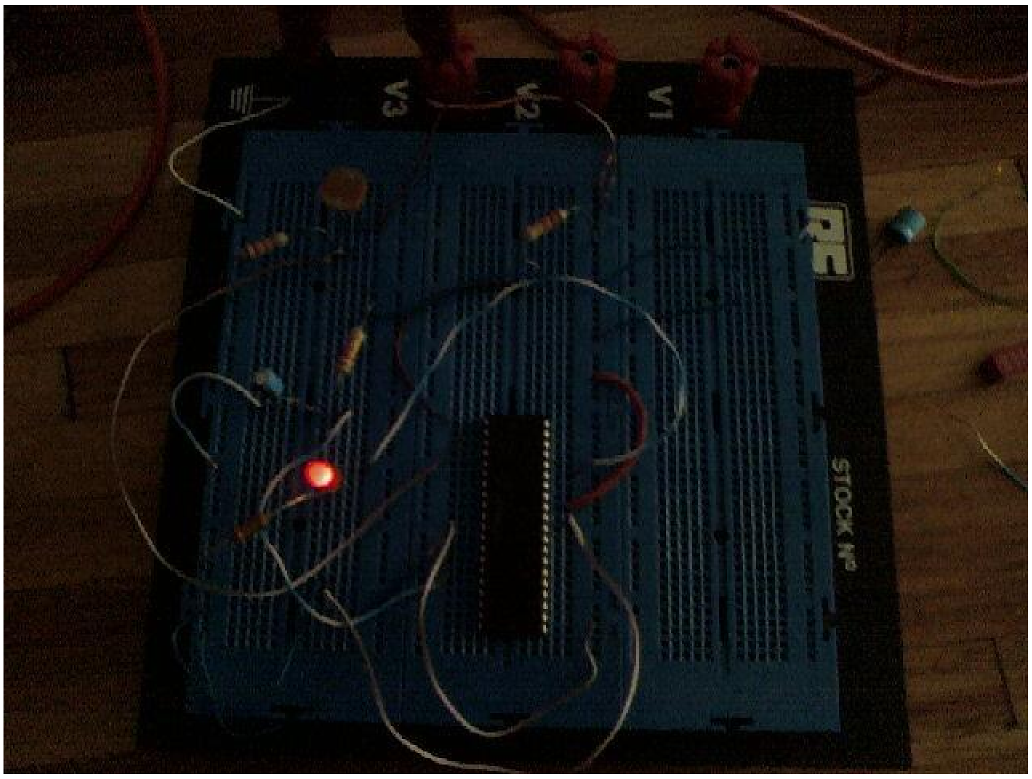


Figure C.2: Light detector circuit when it is dark.

Appendix D

Code for creating computer interface

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class frame extends JFrame {
    // Variables declaration
    private JButton jButton1;
    private JButton jButton2;
    private JButton jButton3;
    private JButton jButton4;
    private JPanel contentPane;
    // End of variables declaration
    public frame()
    {
        super();
        initializeComponent();
        this.setVisible(true);
    }
    private void initializeComponent()
    {
        jButton1 = new JButton();
        jButton2 = new JButton();
        jButton3 = new JButton();
        jButton4 = new JButton();
        contentPane = (JPanel)this.getContentPane();

        // jButton1
        jButton1.setText("Open the blinds");
        jButton1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e)
            {
                jButton1_actionPerformed(e);
            }
        });
        // jButton2
        jButton2.setText("Turn on lights");
        jButton2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e)
            {
                jButton2_actionPerformed(e);
            }
        });
```

```
// jButton3
jButton3.setText("Close the blinds");
jButton3.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        jButton3_actionPerformed(e);
    }
});
// jButton4
jButton4.setText("Turn off lights");
jButton4.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        jButton4_actionPerformed(e);
    }
});
// contentPane
contentPane.setLayout(null);
addComponent(contentPane, jButton1, 26,35,114,48);
addComponent(contentPane, jButton2, 26,152,114,48);
addComponent(contentPane, jButton3, 264,35,114,48);
addComponent(contentPane, jButton4, 264,152,114,48);

// frame
this.setTitle("frame - extends JFrame");
this.setLocation(new Point(164, 41));
```

```
    this.setSize(new Dimension(423, 300));
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
private void addComponent(Container container,Component c,int x,
                           int y,int width,int height)
{
    c.setBounds(x,y,width,height);
    container.add(c);
}
private void jButton1_actionPerformed(ActionEvent e)
{
    JOptionPane.showMessageDialog(null,"Blinds opened here!")
}
private void jButton2_actionPerformed(ActionEvent e)
{
    JOptionPane.showMessageDialog(null,"Lights lit here!")
}
private void jButton3_actionPerformed(ActionEvent e)
{
    System.out.println("\njButton3_actionPerformed(ActionEvent e)
                       called.");
}
private void jButton4_actionPerformed(ActionEvent e)
{
    System.out.println("\njButton4_actionPerformed(ActionEvent e)
                        called.");
    }
        //For Testing
    public static void main(String[] args)
    {
        JFrame.setDefaultLookAndFeelDecorated(true);
        JDialog.setDefaultLookAndFeelDecorated(true);
        try
        {
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.
                                         WindowsLookAndFeel");
        }
        catch (Exception ex)
        {
            System.out.println("Failed loading L&F: ");
            System.out.println(ex);
        }
        new frame();
    }
//= End of Testing =
}
```

Appendix E

Code for checking available ports on a computer

```
import gnu.io.CommPortIdentifier;
import java.util.Enumeration;
public class ListAvailablePorts {
     public void list() {
        Enumeration ports = CommPortIdentifier.getPortIdentifiers();
        while(ports.hasMoreElements())
           System.out.println(((CommPortIdentifier)ports.nextElement())
                             .getName());
        }
        public static void main(String[] args) {
            new ListAvailablePorts().list();
        }
}
```

Appendix F

Serial port data sender

```
import java.io.IOException;
import java.io.OutputStream;

public class CommPortSender {
    static OutputStream out;
    public static void setWriterStream(OutputStream out) {
      CommPortSender.out = out;
    }
    public static void send(byte[] bytes) {
      try{
          System.out.println("SENDING: " + new String(bytes, 0,
                                                      bytes.length));
// sending through serial port is simply writing into OutputStream
          out.write(bytes);
          out.flush();
        }catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Appendix G

Code for Setting serial port connection

```
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort
 public class RS232Example {
    public void connect(String portName) throws Exception {
        CommPortIdentifier portIdentifier = CommPortIdentifier
                            .getPortIdentifier(portName);
        if (portIdentifier.isCurrentlyOwned()) {
            System.out.println("Port in use!");
        } else {
           // points who owns the port and connection timeout
           SerialPort serialPort = (SerialPort) portIdentifier
                            .open("RS232Example", 2000);
           // setup connection parameters
           serialPort.setSerialPortParams(
                38400, SerialPort.DATABITS_8, SerialPort.STOPBITS_1,
                SerialPort.PARITY_NONE);
            // setup serial port writer
            CommPortSender.setWriterStream(serialPort.getOutputStream());
        }
    }
    public static void main(String[] args) throws Exception {
        // connects to the port
        new RS232Example().connect(args[0]);
        // send data 00001111 through serial port
        CommPortSender.send(new byte[] {0x0F});
    }
}
```

Appendix H

Code for Receiving data from RS232

```
list    p=16f887    ; list directive to define processor

#include<p16f887.inc> ; processor specific variable definitions

__config _CONFIG1, _LVP_OFF & _FCMEN_ON & _IESO_OFF & _BOR_OFF &
 _CPD_OFF & _CP_OFF & _MCLRE_ON & _PWRTE_ON & _WDT_OFF &
 _INTRC_OSC_NOCLKOUT
__config _CONFIG2, _WRT_OFF & _BOR21V

CBLOCK 20h   ; Temporary storage
   dc2      ;General purpose registers for storing received data
   dc1
   ENDC

    org     0x0000
    goto Init
```

```
;Subroutines:
 Init
     clrw              ; Zero.
     movwf   PORTB     ; Ensure PORTB is zero before we enable it.
     bsf     STATUS,RP0      ; Select Bank 1
     movlw   b'00000000'     ; Set port B bits as outputs
     movwf   TRISB           ; Set TRISB register.

;*** UART Set Up ***
    MOVLW   0x19            ; 0x19 = 9600 BPS (0x0C = 19200 BPS)
    MOVWF   SPBRG           ; Baud Rate Generator Store Location
    BSF     TXSTA,BRGH      ; High Baud Rate Select bit
    BCF     STATUS,RP0      ; Bank 0
    BSF     RCSTA,SPEN      ; Serial Port Enable bit
    BSF     RCSTA,CREN      ; Continuous Receive Enable bit

Main ;*** Receive *** Receive:
    BTFSS   PIR1,RCIF    ; P1R1 is set when byte is recieved
    GOTO    Receive
    MOVF    RCREG,W       ; Move received data into W
    movwf   dc2  ; move data from the working register into general
    btfss   dc2,1  ; purpose register & check if the  second bit is set
    goto   blind_control   ; if clear then go to blind control
    goto   light_control   ; if it is set then go to light control
```

```
open_blind
    movlw b'1001' ;90 degrees clockwie
    movwf PORTB
    call wait
    goto Receive

close_blind
    movlw b'0110'  ;90 degrees anticlockwise
    movwf PORTB
    call wait
    goto Receive

turn_on_lights
    bsf PORTB,2
    goto Receive

blind_control
     btfss dc2,0         ;check if the first bit is set or clear
     goto open_blind     ;if clear
     goto close_blind    ;if set

light_control
     btfss dc2,0          ;check if the first bit is set or clear
     goto turn_on_lights  ;if clear
     bcf PORTB,2         ;if set then turn off lights
     goto Receive
```

```
wait
      movlw   5   ; Outer loop iteration count
d11   movwf   dc1

    decfsz  dc1,F  ;Decrement 5, Skip if 0
    goto    $-1   ;wait for loop count to finish
    return
 end
```

Appendix I

Photographs of Serial Communication

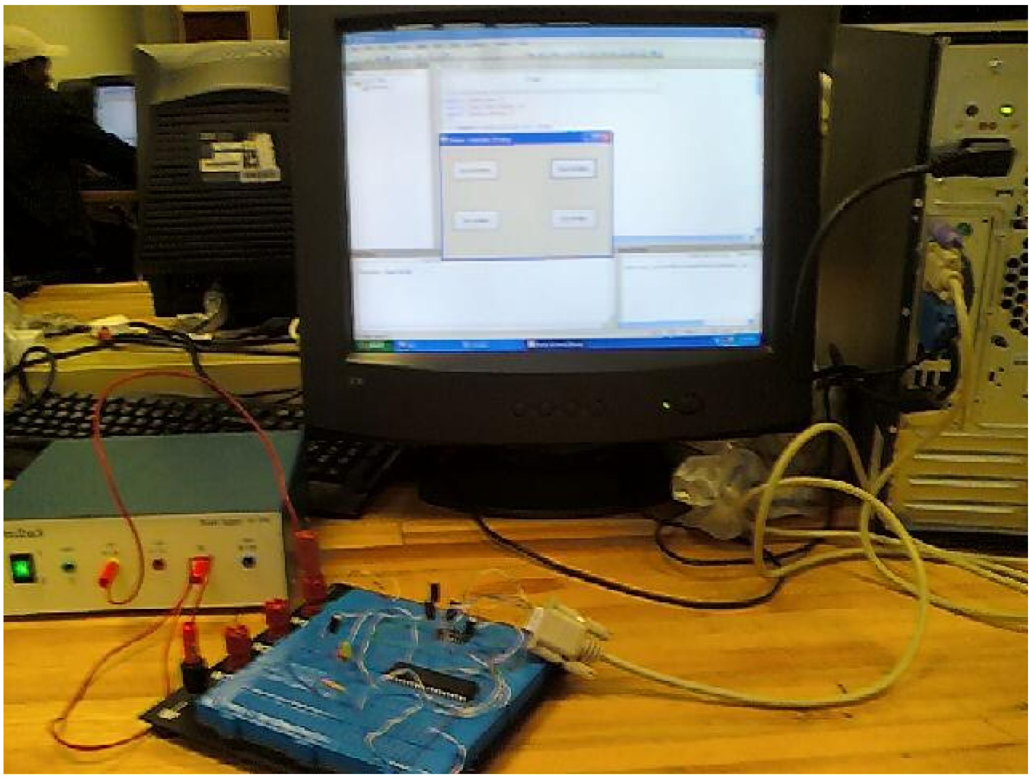


Figure I.1: Serial Communication.

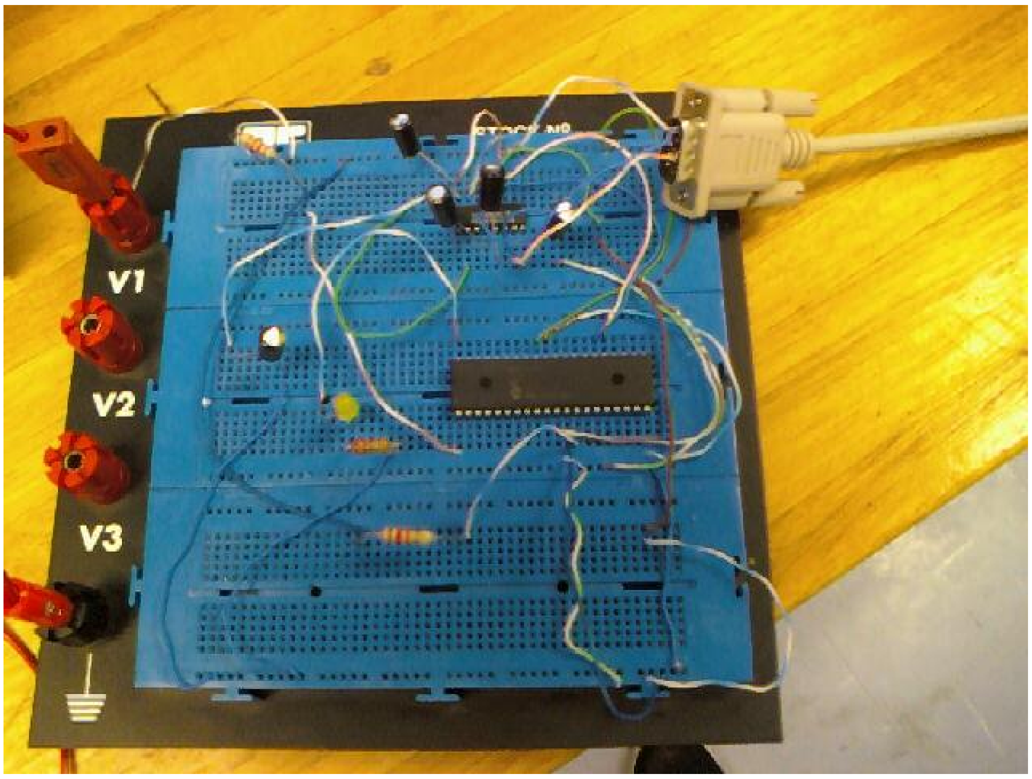


Figure I.2: Close view to braid board.

Appendix J

Cost Analysis

Component	Description	Quantity	Cost
Microcontroller	PIC16F887	1	M25.36
Light sensor	LDR	1	M28.27
Motor Driver	ULN2803AN	1	M8.03
Motion sensor	Infrared photo detector	1	M28.38
Max232		1	M39.62
OP Amplifier	LM358P	1	M7.03
Capacitor	100uF	5	M8.45
	10uF	5	M10.90
Transistor	2N9304	5	M4.32
Motor	Unipolar stepper motor	1	M432.96
serial cable	RS232	1	M50.74
Resistor	1kΩ	3	M2.45
	12kΩ	3	M3.00
	330Ω	3	M2.45
Circuit board	braid board	3	from Lab
Wires			from Lab
LED		5	from Lab
Total cost			**M651.96**

Table J.1: Cost Analysis.

Appendix K

Complete code of automatic light control system for Smart homes

```
;PROGRAM FUNCTION:light control

list    p=16f887    ; list directive to define processor

#include<p16f887.inc>  ; processor specific variable definitions

 __config _CONFIG1, _LVP_OFF & _FCMEN_ON & _IESO_OFF
  & _BOR_OFF & _CPD_OFF & _CP_OFF & _MCLRE_ON & _PWRTE_ON
  & _WDT_OFF & _INTRC_OSC_NOCLKOUT
  __config _CONFIG2, _WRT_OFF & _BOR21V

;Declarations:
 NumH     equ   20h
 NumL     equ   21h
 Mark241  equ   22h
 NumH1    equ   23h
 NumL1    equ   24h
 PostX    equ   25h
 Mark240  equ   26h
 PostY    equ   27h
 Counter8 equ   28h
 dc1      equ   29h

    org     0x0000
    goto Start

Init
;-----------------------------------------------------------------------
  ;Initializing porta
    BCF STATUS, RP0 ;
    BCF STATUS, RP1 ; Bank0
    CLRF PORTA      ; Initialize PORTA clearing output data latches
    BSF STATUS, RP0 ; Select Bank 1
    MOVLW 0x06      ; Configure all pinss
    MOVWF ADCON1    ; as digital inputs
    MOVLW 0xCF      ; Value used to initialize data  direction
    MOVWF TRISA     ; Set RA<3:0> as inputs RA<5:4> as outputs
                    ;TRISA<7:6>are always read as ’0’
 ; Initializing portb
    clrw                    ; zerp
    movwf   PORTB           ; Ensure PORTB is zero before we enable it.
    bsf     STATUS,RP0       ; Select Bank 1
    movlw   b'00000000'      ; Set port B bits as outputs
    movwf   TRISB            ; Set TRISB register.
    bcf     STATUS,RP0       ; Select Bank 0

   movlw    b'00000111' ;sets up timing register
   option

   ;initializing ADC for light sensor
;Set ADCON0
    movlw   b'10000001'   ; FOSC/32,RA0/AN0,ADON=ON
    movwf ADCON0
;Set ADCON1
    BANKSEL ADCON1        ; Change to bank 1
    movlw   b'10000000'  ;Right justified,VSS,VDD
    movwf   ADCON1
    BANKSEL ADCON0        ;Change back to bank 0

  ;initializing ADC for motion sensor
;Set ADCON0
    movlw   b'10000101'; FOSC/32,RA1/AN1,ADON=ON
    movwf   ADCON0 ;
;Set ADCON1
    BANKSEL ADCON1    ; Change to bank 1
    movlw   b'10000001' ;;Right justified,VSS,VDD
    movwf   ADCON1
    BANKSEL ADCON0  ;Change back to bank 0
    return
;------------------------------------------------------------------------
;Subroutines: is_it_dull
    ; Read_ADC for light sensor
    bsf ADCON0, GO_DONE     ;initiate conversion
    btfsc   ADCON0, GO_DONE ;Check for end of conversion
    goto    $-1             ;wait for ADC to finish
    movf    ADRESH,W       ;store ADRESH value into working register
    andlw   0x03          ;select two bits
    movwf   NumH          ;store in register NumH
    BANKSEL ADRESL        ;Change to bank 1
    movf    ADRESL,W      ;store ADRESL value into working register
    BANKSEL ADRESH        ;change to bank 0
    movwf   NumL         ;return result in NumL and NumH
    btfss NumH,0
    goto fully_open_blinds

slightly_open_blinds
    movlw b'1011'  ; turn motor 45 degree
    movwf PORTB
    call wait
    bcf PORTB,7  ;turn off LED
    goto Main
;------------------------------------------------------------------------
motion_2
```

```
        ;Read ADC for motion sensor
        bsf  ADCON0, GO_DONE     ;initiate conversion
        btfsc   ADCON0, GO_DONE
        goto    $-1           ;wait for ADC to finish
        movf    ADRESH,W    ;store ADRESH value into working register
        andlw   0x03       ;select two bits
        movwf   NumH1      ;store in register NumH1
        BANKSEL ADRESL      ;Change to bank 1
        movf    ADRESL,W    ;store ADRESL value into working register
        BANKSEL  ADRESH     ;change to bank 0
        movwf    NumL1   ;return result in NumL1 and NumH1
        return
;------------------------------------------------------------------------
motion_1
        ;Read ADC for motion sensor
        bsf  ADCON0, GO_DONE     ;initiate conversion
        btfsc   ADCON0, GO_DONE
        goto    $-1           ;wait for ADC to finish
        movf    ADRESH,W    ;store ADRESH value into working register
        andlw   0x03       ;select two bits
        movwf   NumH1      ;store in register NumH1
        BANKSEL ADRESL      ;Change to bank 1
        movf    ADRESL,W    ;store ADRESL value into working register
        BANKSEL  ADRESH     ;change to bank 0
        movwf    NumL1   ;return result in NumL1 and NumH1
       ;Check if there is any motion detected
```

```
      btfss NumH1,1
      goto led_oon
      bsf PORTB,7       ;turn on the LED
      goto Main
led_oon
      bsf PORTB,7       ;turn on the LED
_4hr_delay
      movlw d'144000'
      movwf PostX       ;sets up variable postscaler
      movlw d'240'      ;sets up fixed marker
      movwf Mark240     ;
TimeLoop
      movfw Mark240     ;waits for TMR0 to count up
      subwf TMR0,w      ;240 times
      btfss STATUS,Z    ;
      goto  TimeLoop    ;hasn't, so keeps looping
      movlw d'240'      ;resets Mark240
      addwf Mark240,f   ;
      decfsz    PostX,f     ;does this X times
      goto  TimeLoop    ;
      bcf     PORTB,7       ;turn off the LED
      movlw d'8'        ;sets up Counter8 with an initial
      movwf Counter8    ;value of 8
      goto _8hr_loop
;---------------------------------------------------------------------
wait      movlw   5   ; Outer loop iteration count d11
```

```
         movwf   dc1
d11
          decfsz  dc1,F
          goto    $-1
          return
;-----------------------------------------------------------------------
timedelay
        movwf   PostY       ;sets up variable postscaler
        movlw   d'240'      ;sets up fixed marker
        movwf   Mark241     ;
Time
        movfw   Mark241     ;waits for TMR0 to count up
        subwf   TMR0,w      ;240 times
        btfss   STATUS,Z    ;
        goto    Time    ;hasn't, so keeps looping
        movlw   d'240'      ;resets Mark240
        addwf   Mark241,f   ;
        goto    motion_2    ;keep on checking if motion is detected
        goto    check      ; to perform required tasks
dky
        decfsz  PostY,f     ;does this X times
        goto    Time    ;
        retlw   0       ;returns after required time
;-----------------------------------------------------------------------
_8hr_loop
       movlw    d'36000' ;sends message of 1 hour to sub;
       call timedelay     ;creates delay of required time
       decfsz   Counter8,f ; runs through this loop 8 times;
       goto _8hr_loop
       goto Main       ;loops back to start
;-----------------------------------------------------------------------
check
        btfss NumH1,1   ;check if motion is detected
        goto ledoff
        bsf PORTB,7  ;turn LED on
        goto dky     ;
ledoff
        bcf PORTB,7  ;turn LED off
        goto dky
;-----------------------------------------------------------------------
Fully_close_blinds
       movlw b'0000'  ;turns coils to original place
       movwf PORTB
       call wait
       goto motion_1    ;check if there is motion detected
;-----------------------------------------------------------------------
fully_open_blinds
       movlw b'1001' ;turn motor 90 degree
       movwf PORTB
       call wait
       bsf PORTB,7   ;turn LED on
       goto Main
```

```
;-------------------------------------------------------------------------
;Program Start:
 Start
    call Init             ;sets the values of ADCON0 and ADCON1
                                        ;initialize porta
Main
;-------------------------------------------------------------------------
Light_sensor
    ; Read_ADC for light sensor
    bsf ADCON0, GO_DONE     ;initiate conversion
    btfsc   ADCON0, GO_DONE ;Check for end of conversion
    goto    $-1            ;wait for ADC to finish
    movf    ADRESH,W       ;store ADRESH value into working register
    andlw   0x03          ;select two bits
    movwf   NumH          ;store in register NumH
    BANKSEL ADRESL        ;Change to bank 1
    movf    ADRESL,W      ;store ADRESL value into working register
    BANKSEL ADRESH        ;change to bank 0
    movwf   NumL         ;return result in NumL and NumH
;-------------------------------------------------------------------------
    ;Check if it is dark or bright
    btfss NumH,1   ;check if output voltage is greater than 2.5v
    goto Fully_close_blinds
    goto is_it_dull
       END
```

Bibliography

[1] Aldrich F.K, *Smart Homes: Past Present and Future*, London: Springer-Verlag UK, 2003.

[2] Martin Edge, Bruce Tayloor and Guy Dewsbury, *The Potential for Smart Home Systems in Meeting the Care Needs of Older People with Disabillities*, The Robert Gordon University, School of Construction, Aberdeen, Scotland, 2000.

[3] Kenneth S. Hurst, *Engineering Design Principles*, University of Hull, John Wiley and Sons Inc., 605 Third Avenue, New York, NY 10158-0012, 2004.

[4] John G. Webster, David Beams, *Measurement, Instrumentation, and Sensors Handbook CRCnetBase*, CRC Press LLC, 1999.

[5] Derci Felix da Silva, Daniel Acosta-Avalos, *Light Dependent Resistance as a Sensor in Spectroscopy Setups Using Pulsed Light and Compared with Electret Microphones*, Instituto de Pesquisa e Desenvolvimento (IPD), Universidade do Vale do Paraba, Brasil, 9 May, 2006.

[6] Larry K. Baxter, *Capacitive Sensors*, IEEE Press, Piscataway N.J., 1997

[7] Arthur Harrison, *Unipolar Stepper Motor Control Circuit* Department of Electrical and Computer Engineering and Coordinated Science Laboratory, University of Illinois, 23 March, 2003

[8] Douglas W.Jones, *Stepping Motors*, University of Lowa, Department of Computer Science, 2005.

[9] *Darington Transistor Array*, Texas Instruments Incorporated, POST OFFICE BOX 655303 DALLAS, TEXAS 75265, August, 2004.

[10] Jan Axelson, *The Microcontroller Idea Book*, 5310 Chinook Ln. Madison, WI 53704 USA, 1997.

[11] Yesu Thommandru, *Programming a PIC Microcontroller*, Iowa State University ECpE, November, 2006.

[12] Microchip, *PIC16F882/883/884/886/887 Data Sheet*, Microchip Technology Inc. 2008.

[13] Elizabeth Gregory, *Serial Port Communication*, National Instruments ,The Connexions Project, Creative Commons Attribution License, 2004.

[14] Moreno, Ivan, Ching-Cherng, *Modeling the radiation pattern of LEDs*, Optics Express, 2008.

[15] Frank Durda, *Serial and UART Tutorial*, Uhclem-FreeBSD.org, January, 1996.