MKEM 1863 - DESIGN OF MICROPROCESSOR - BASED MECHATRONIC SYSTEM

ROBOT DESIGN

PROJECT REPORT

OSAMA MAZHAR

PREPARED FOR:

ASSOC. PROF. DR. ROSBI BIN MAMAT

UNIVERSITI TEKNOLOGI MALAYSIA

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVE:

The objective assigned by the lecturer for this project is as follows:

"The task is to design a robot with appropriate sensors and software to navigate through the grid to detect and collect data. The squares with high acid level must be remembered for later reporting. The robot will be based on existing frame with 2 servos controlling the wheels."

1.2 OVERALL STRATEGY:

The robot is supposed to follow the outer grid-line on the right extreme, and scans for any acidic/dark patches in the grid-blocks using a sensor mounted on an extension towards left-side. The length of the extension is such that the sensor comes in the middle of the grid-block. The sensor mounted on the extension will be used for detecting the position and number of the patches.

The project is divided into three main parts as follows:

1. Locomotion of the robot in the grid
2. Identifying the blocks/grid-lines
3. Detecting the acidic/dark patches

For each part the description of the problem and its respective solution is discussed in the following sections.

1.2.1 LOCOMOTION OF THE ROBOT

The task of detecting the acidic/dark patches starts with the locomotion of the robot i.e. the robot must go inside the grid, scan the individual blocks and detect for acidic patches. The robot consists of two servo motors that control the wheels. It is a common observation that the servos may not be identical in operation to each other. It is because that they may be from different manufacturer, or from a different batch from the same manufacturer, or the performance of either of the motor does not match with the other due to wear and tear. Therefore it is assumed that the robot may not go into the straight line if only the motors are programmed with the same input values for the pulse width.

In order to keep the robot moving in a straight line so that it does not go outside the grid, the group decided to work on a line following robot and use the grid lines as the reference lines for the robot.

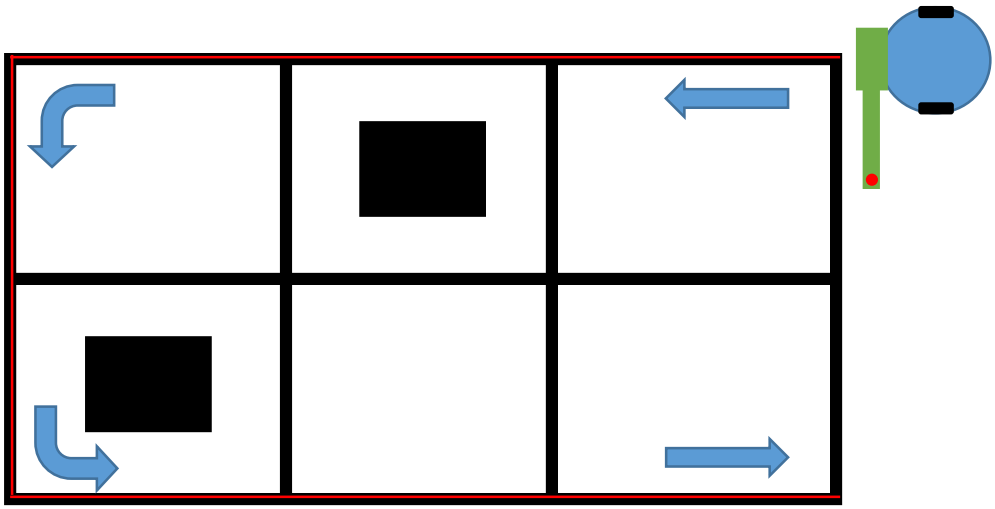

Figure 1-1 Illustration of the grid and overall strategy of the robot

Searching for the availability of the sensors for the line following robot, a self-calibrating sensor with five infra-red opto-couplers named as LSS05 was found on the Cytron website as shown in the following figure.

Figure 1-2 LSS05 auto-calibration line following sensor

1.2.1.1 Auto-calibration of the sensor:

LSS05 sensor uses a PIC microcontroller to calibrate the thresholds of the ir-receivers so that it can provide binary output to the circuit. The calibration can be either performed by a push button present on the sensor or by software using a hardware pin provided in the connector.

The calibration is done such that when the push button is pressed, the PIC microcontroller starts storing the minimum and maximum values from the input that it receives from the ir-detector. The

user is asked to move the sensor over line and background for about three seconds and the thresholds for the line and the background are set in the EEPROM of the microcontroller. Later those threshold values are used to provide the binary output to the main-board of the robot.

1.2.1.2 Line following algorithm:

For line following an alogorithm is designed which works on the difference values. Following picture shows the concept of the difference algorithm for the line following with several possibilities of combinations while the robot is following the line.

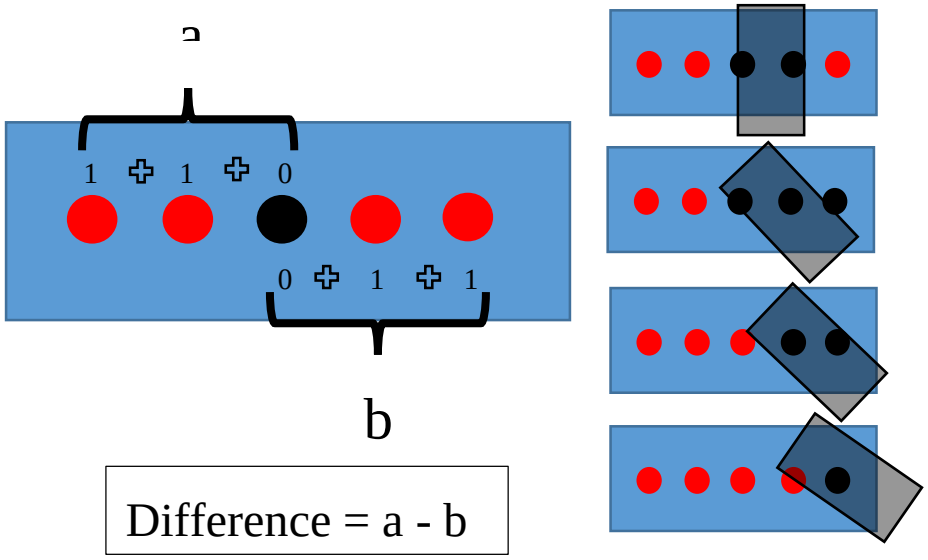

Figure 1-3 Concept of line-following algorithm

For the difference equals to zero means that the robot has to move in the straight line. Whereas inclination of the difference values towards higher positive or negative values will determine how robot should react to make difference zero again.

1.2.2 IDENTIFICATION OF THE BLOCKS/GRID-LINES:

For the purpose of detecting the blocks the grid-lines separating the blocks are used to increment the count for the blocks. As the robot moves along the outer grid as shown in the previous section and the LSS05 is used to follow the lines, it was decided to introduce another infra-red opto-coupler to detect the grid-lines. The final setting of the sensor hardware is shown below:

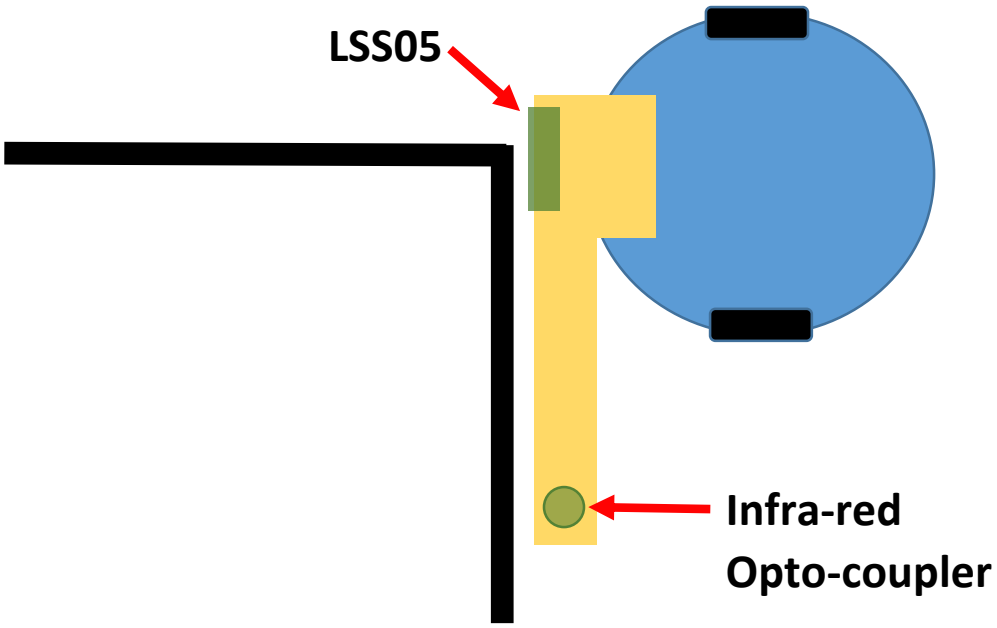

Figure 1-4 Position of the robot and hardware setup for grid-line and patch detection

1.2.2.1 Hardware of infra-red sensor:

The infra-red coupler is connected in a voltage divider circuit and its output is fed into the analog input of the Arduino. The circuit for infra-red detector is shown in the following figure:

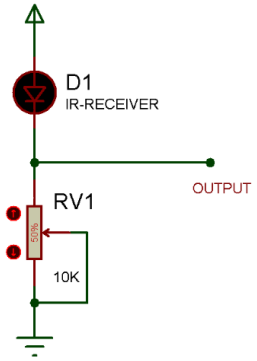

Figure 1-5 Voltage divider circuit for ir-sensor

The output of the voltage divider is fed into sixth analog input (A5) of Arduino UNO board. The digital value equivalent of the voltage output is analyzed experimentally to see the threshold for the detection of black grid lines. The threshold is set in the program and a function is written to convert the output of the infra-red detector to binary 1 and 0 output with respect to the threshold set for the line and background.

1.2.2.2 Strategy for detecting blocks/gridlines:

Furthermore counter technique is used for the detection of grid lines. As soon as the ir-sensor comes over the grid line it gives a HIGH signal as discussed above. The program is then forwarded to a loop which runs until the ir-sensor is over the grid-line. This loop repeatedly increments a variable until the ir-sensor crosses the line and the output of the sensor returns to LOW again. A band for the counts is defined using experiments having a lower and upper limit for the detection of grid-lines. As long as the count remains in the pre-defined band, detection of grid-line is inferred. If the count value is less than or greater than the lower and upper limit respectively the microcontroller does not infer detection of a grid-line, it takes another decision and moves on.

1.2.3 DETECTION OF THE ACIDIC/DARK PATCHES:

Patch detection is similar to the detection of grid-lines. The following figure shows the difference in time during which the ir-sensor senses the black color representing either a grid-line or a patch. It is obvious from the figure that ir-sensor remains for a larger period of time over the patch as compared to that over the gird-line.

1.2.3.1 Strategy of patch detection:

The same counter variable is observed to get number greater than the upper limit of the band for the detection of grid-lines. If the number is greater than the upper limit of the set band, detection of a patch is inferred respectively.

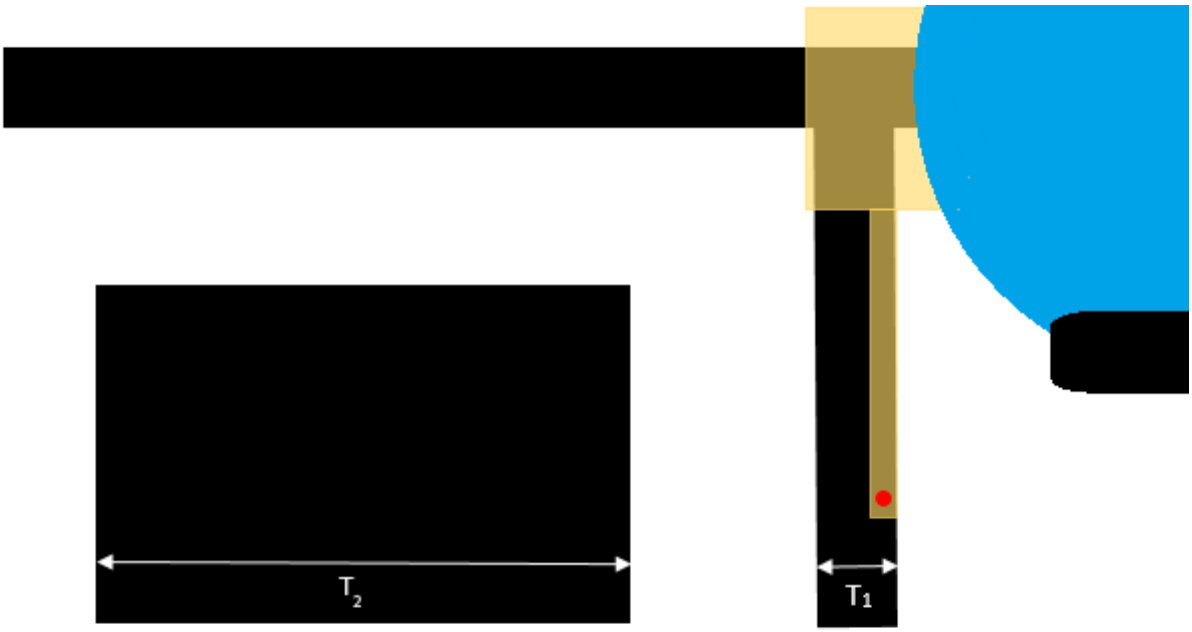

Figure 1-6 Illustration of the strategy of grid-line and patch detection

CHAPTER 2

HARDWARE OF THE ROBOT

2.1 OVERALL DESIGN OF THE ROBOT:

The following figure shows an overall design of the robot hardware. The base of the robot has two continuous servos mounted over the board. Two round-headed nut mounted on the two screws on the two extremes of the robot acts as caster wheels/support. Sensor attachment, Arduino UNO board, 16X2 LCD Matrix, Battery and a power switch was mounted over the robot as required.

`

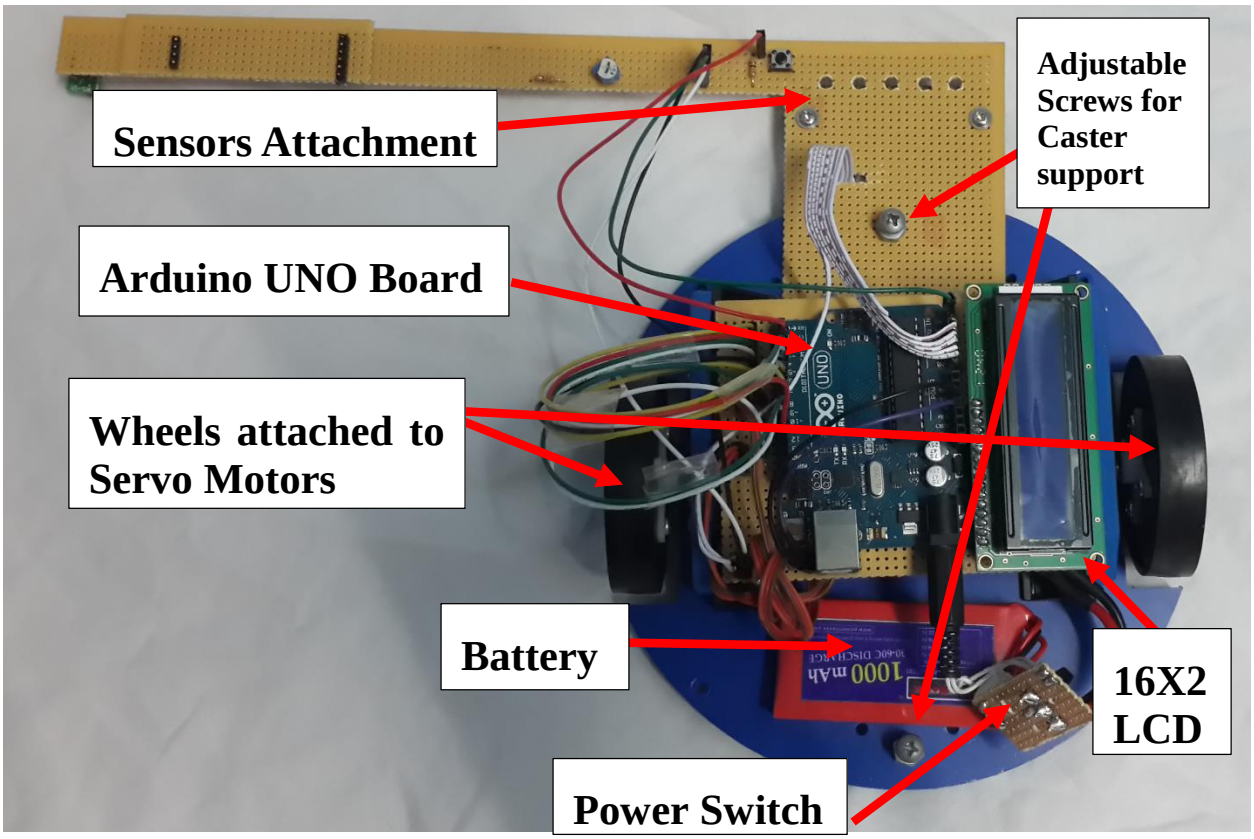

Figure 2-1 Overall structure of the robot

2.1.1 SENSORS MOUNTING:

The mounting of the sensors is fabricated using a strip-board. An extension for the ir-sensor is made for the detection of grid-lines and acidic patches. The Arduino board is mounted over the robot using a double sided tape. The connections are soldered on the main strip-board and some are made using jumper-wires as shown in the figure.

CHAPTER 3

PROGRAMMING OF THE ROBOT

3.1 PROGRAMMING OF SERVO MOTORS:

Servo motors with feedback potentiometer are allowed to move from 0^O to 180^O using a <Servo.h> header file. The continuous servos are programmed such that they are halted when input is 90^O. They rotate in one direction when the input is increased from 90^O to 180^O and the motors rotate in the opposite direction when the input is less than 90^O.

1. The servo object has to be defined using a command as follows:

   Servo *servoname*

2. Later the pin-out for the pulse output to the motors has to be defined as follows:

   Servoname.attach(*pinnumber);*

   Only specific pins with analog (PWM) output can be used to drive servo motors. Those pins are 3, 5, 6, 9, 10, and 11.

3. In order to move servos the following command is used:

   Servoname.write(*argument from 0-180);*

As the argument moves farther from 90 the speed of the servo increases in one direction and vice-versa. The following picture shows the mounting of the servos on the robot:

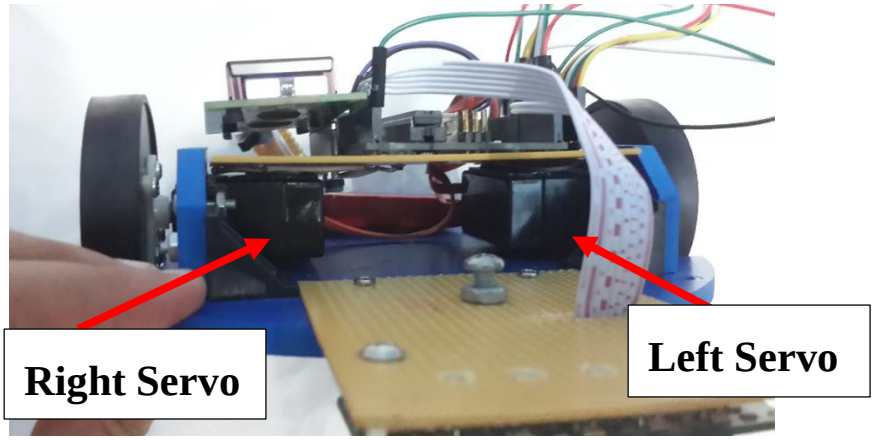

Figure 3-1 Mounting of servo motors on the robot

3.2 PROGRAMMING OF 16X2 DOT-MATRIX LCD:

There are two modes for a 16X2 dot-matrix LCD operation namely 8-bit mode and 4-bit mode. A 16X2 LCD based on Hitachi HD44780 or compatible can be programmed using <LiquidCrystal.h> library/header. The LCD has hardware pins as shown in the following figure:

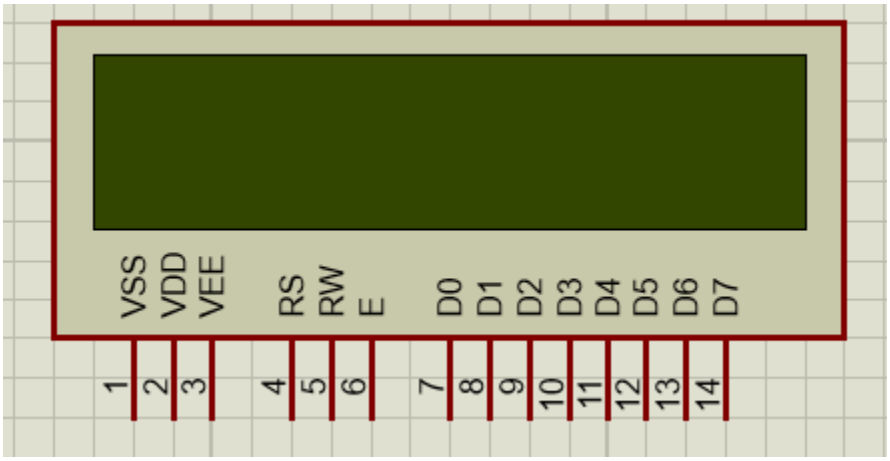

Figure 3-2 A schematic of a 16X2 dot-matrix LCD

VSS, VDD and VEE are used to power LCD and set its contrast. RS (register select), RW (read/write), and E (enable) pins are control pins that control the operation of LCD. RS pin is used for selecting either command or data register, RW is used to direct read/write operation from and to LCD and E pin is used to strobe LCD to execute an operation either reading/writing data or command. Remaining 8-bits from D0 to D7 are known as data pins. They are used to write/read 8-bit data (ASCII) or command to the LCD. These LCDs may have two more pins for backlight ON/OFF.

If all the pins are used and connected to microcontroller, the operation is known as 8-bit operation which is faster and dependent on the flag feedback from D7. If 4-data pins are used to send data/command (in nibbles) and only RS and E pins are used to control the operation then the operation is known as 4-bit which does not require any flag feedback but it is a slower operation as compared to 8-bit operation. This is so because the 8-bit data is sent to LCD in nibbles which require almost double the time when compared to 8-bit operation and since no flag-bit is used thus the program needs to wait for a specified time for the LCD to complete its operation well before reading another data/command from the microcontroller.

Following are some commands that are used to display messages/numbers on the LCD:

LiquidCrystal lcd(*RS, E, D4, D5, D6, D7*); // To initialize LCD in the program

lcd.print(*byte or a string*); // String must be in double quotes

3.3 READING ANALOG DATA:

Analog data can be read using analog inputs of the Arduino. In this project only the data from ir-sensor mounted on the extension is read through analog channel A5 (6^{th} analog input pin). The following command is used to read analog data and store it into an integer:

Integer = analogRead(*channel name from A0-A5*);

3.4 OVERALL PROGRAMMING STRATEGY AND DESCRIPTION:

The overall programming description is discussed in the following sections. The sections are broken as the initializing and data transfer part, normal robot operation for patch scanning and displaying the data after the robot finishes the scan. Each part is explained briefly in a numbered bullet fashion as follows:

3.4.1 INITIALIZING AND DATA TRANSFER:

1. The program starts by initializing the objects and variables used in the robot.
2. It then enters a loop scanning the digital pin for push-button input from the user and displays a message for the user to press a button.
3. As soon as the user press the push-button connected to pin-number 2 of Arduino board the program scans until 2 seconds for the button still pressed or released.
4. If the button is still pressed, it displays a message to either release the button within 1 second or keep pressing it.
5. If the user release the button within 1 second the program enters into another loop sending the already stored data from EEPROM to LCD as well as to the serial port.
6. If the button was pressed until after 1 second passed the program starts its function as normal.

3.4.2 PATCH SCANNING OPERATION:

1. The robot starts with a line-follow function and scans for any signal at ir-sensor mounted on the extension.
2. If there is no signal at the ir-sensor which means that no grid-line dividing the block appeared neither the patch, the robot will keep following the outer boundary of the grid.
3. If the robot receives a signal from ir-sensor, it will enter a loop which commands the robot to keep follow the line but starts a counter and increment its value until ir-sensor is receiving a signal.
4. As soon as the signal is over, the robot compares the count value with a pre-defined band with upper and lower limit of the count which is set after successive experiments.
5. If the count value is inside the set band for the grid-line, it increments a variable representing grid-line and displays it on the LCD.

6. If the count value is greater than the set band then it increments the variable representing patch number and writes in a respective address of EEPROM for storing the position of the patch.
7. If the count value is less than the set band, the value is dis-regarded and assumed to be noise.
8. The robot keeps on moving ahead until the grid-line value equals 4 which means that the robot needs to take a left-turn until it finds the line again.
9. The value for grid-line is incremented in the program to 5 so that it does not turn again when the program reaches that line in the second cycle.
10. It keeps moving and scans for any patch until its grid-line value reaches 7, which means that the robot needs to take another left turn and the grid-line value is incremented to 8 after it has taken the turn.
11. The robot keeps moving and scans for the patches, storing the data in respective EEPROM addresses until the grid-line value equals 11 which means that the scanning is complete.
12. The robot goes straight for 1 second and then stops and goes into the next loop.

3.4.3 DATA DISPLAY LOOP:

1. The robot waits for the user input to proceed further.
2. It gets all the patch values from EEPROM and store them in an array including the value for total number of patches.
3. The program enters an infinite loop for displaying the data/result on LCD.
4. The program ends.

APPENDIX A

4.1 SCHEMATIC DRAWING OF ROBOT ELECTRONICS

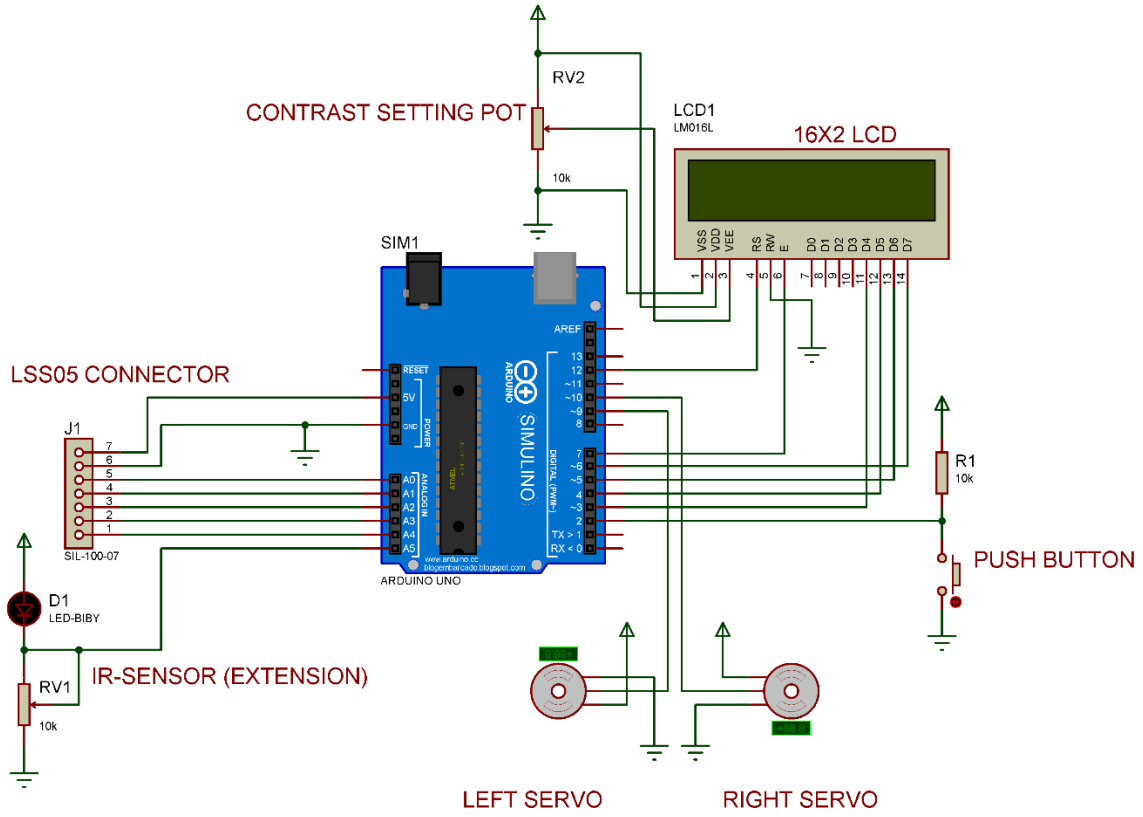

Figure 4-1 Schematic diagram of robot electronics

APPENDIX B

5.1 ESTIMATED COST

S/N	ITEM DESCRIPTION	COST
1.	Arduino UNO	RM 101.76
2.	LSS05 (Auto-Calibration)	RM 58.30
3.	16×2 LCD	RM 19.08
4.	IR-Sensor Set (for patch)	RM 4.77
5.	Strip Board	RM 4.03
6.	Jumper Wires	RM 4.77
7.	Soldering Wire	RM 3.18
8.	Push Button, Power Connector and Misc.	RM 4.00
TOTAL COST	**----------**	**RM 199.89**

Note:

1. The above cost estimate excludes the cost of servo motors, robot base and the battery.
2. All the required items were bought from cytron store, and the cost mentioned is accordingly for their website (www.cytron.my)