|  |
| --- |
| **18EEC308J-Embedded System Design (LAB)** |
| **RECORD**  **SEMESTER VI ACADEMIC YEAR: 2021-22**  **NAME: Akarsh Agarwal. REG. NO.: RA1911044010013**    **DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING**  **FACULTY OF ENGINEERING & TECHNOLOGY**  **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  (Formerly SRM University, Under section 3 of UGC Act, 1956)  **S.R.M. NAGAR, KATTANKULATHUR – 603 203 KANCHEEPURAM DISTRICT** |



SRM Institute of Science and Technology

(Deemed to be University)

###### S.R.M. NAGAR, KATTANKULATHUR -603 203 KANCHEEPURAM DISTRICT

**BONAFIDE CERTIFICATE**

**Register No** RA1911044010013

Certified to be the bonafide record of work done by Rijo Saju Verghese of Electrical and Electronics department, B.Tech degree course in the Practical of 18EEC308J Embedded System Design in **SRM IST, Kattankulathur** during the academic year 2021-2022.

**Lab in-charge**

###### Date: Year Co-ordinator

Submitted for end semester examination held in Lab, SRM IST**,** Kattankulathur**.**

###### Date: Examiner-1 Examiner-2

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.**  **No.** | **Name of the Experiment** | **Marks (50)** | **Signature of the Staff** |
| 1 | Basic Assembly Language Programming. |  |  |
| 2 | LED Blinking using 8051 Microcontroller. |  |  |
| 3 | PWM-Wave Generation |  |  |
| 4 | Line Follower Robot using Arduino. |  |  |
| 5 | Arduino based Sun Tracking Solar Panel. |  |  |
| 6 | Traffic Light Control using ATMega250. |  |  |
| 7 | DC Motor Speed Control Using Arduino. |  |  |
| 8 | Automatic water level indicator and Controller using  Arduino. |  |  |

**DEPT. OF ELECTRICAL & ELECTRONICS ENGINEERING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, Kattankulathur – 603203.**

|  |
| --- |
| Title of Experiment : Basic Assembly Language Programming |
| Name of the candidate : Akarsh Agarwal  Register Number : RA1911044010013  Date of Experiment : 10/01/2022  Date of submission **:** |

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO:** | **MARKS SPLIT UP** | **MAXIMUM MARKS (50)** | **MARKS OBTAINED** |
| 1 | PRE LAB | 5 |  |
| 2 | PROGRAM | 25 |  |
| 3 | EXECUTION | 15 |  |
| 4 | POST LAB | 5 |  |
| TOTAL | | 50 |  |

**Staff Signature**

###### PRE-LAB

1. **Specify the number of registers in a 2K memory chip?**

Ans: A 2K memory chip has 2048 bits or registers.

###### What is an assembler?

Ans: An **assembler** is a program used to convert or translate programs written in assembly code by humans to machine code (binary) that can be understood by the computer.

###### What are the advantages of an assembly language in comparison with high level language?

Ans: Performance and accuracy are better than high-level language. Executable code is less than high-level language thus takes less time to execute & program runs faster.

###### List the components of computer?

Ans: A motherboard

A Central Processing Unit (CPU)

A Graphics Processing Unit (GPU), also known as a video card Random Access Memory (RAM), also known as volatile memory Storage: Solid State Drive (SSD) or Hard Disk Drive (HDD)

###### What is an operating system?

Ans: An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

* 1. **Basic Assembly Language Programming**

**Aim:**

To do the arithmetic operations using 8051 microprocessors

**Apparatus required:**

**Hardware Requirement :**

8051 Microcontroller kit, Power supply

**Software Requirement:**

8051 EdSim

**Algorithm:**

**Addition / Subtraction**

Step 1 : Move 1H data to memory

Step 2 : Add or subtract 1H data with 2nd data Step 3 : Initialize data pointer.

Step 4 : Move result to memory pointed by DPTR.



START

Out 1H data in memory

Add or subtract 1H and 1st data



Stop

Move result to memory preset by DPTR

Initialize DPTR

**Multiplication / Division**

Step 1 : Get 1H data and 2nd data to memory Step 2 : Multiply or divide 1H data with 2nd data Step 3 : Initialize data pointer.

Step 4 : Move result to memory pointed by DPTR (first port) Step 5 : Increment DPTR

Step 6 : Move 2nd part of result to register A

Step 7 : Move result to 2nd memory location pointer by DPTR





START

Get data into the register

Complement the data

Move the data to pointer by DPTR



Increment data

Increment DPTR



Move data into paste location



Stop

Short jump to preset location

**Program: 8-bit Addition:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Memory Location** | **Label** | **Opcode** | **Mnemonics** | **Comments** |
| 4100 |  | 74 01 | MOV A, #01 | Moves data 1 to  register A |
| 4102 |  | 24 02 | ADD A, #02 | Add content of A and data 2 and store in A |
| 4104 |  | 90 45 00 | MOV DPTR,#4500 | Moves data 4500 to DPTR |
| 4107 |  | F0 | MOVX @DPTR,A | Moves control of A to location pointed DTPR |
| 4108 |  | 80 FE | SJMP 4108 | Short jump to 4108 |

**Execution Addition**

|  |  |
| --- | --- |
| ML | Input |
| 4101 |  |
| 4103 |  |

|  |  |
| --- | --- |
| ML | Output |
| 4500 |  |
| ML | Output |
| 4500 |  |

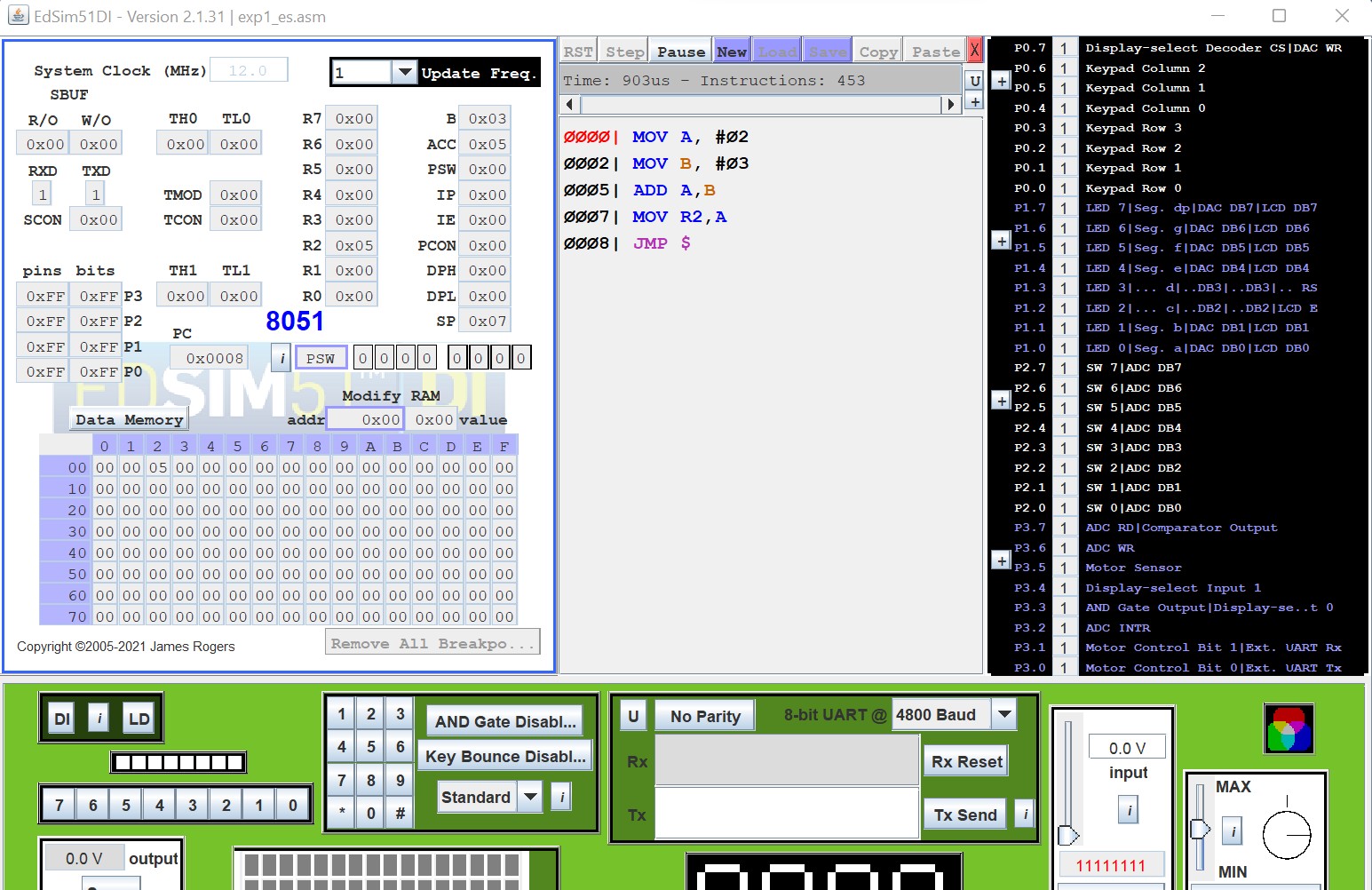
**EDSIM51 PROGRAM-ADDITION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ADDRESS** | **LABEL** | **MNEMONICS** | **OPCODE** | **COMMENTS** |
|  |  | MOV A, #Data |  | Move a data (8 bit) to Acc |
|  |  | MOV B, #Data |  | Move a data (8 bit) to B Reg |
|  |  | ADD A, B |  | Add them, and store result in A |
|  |  | MOV R2, A |  | Copy the result to a Reg |
|  |  | JMP $ |  | Jump to Address location |

|  |  |
| --- | --- |
| **IN PUT ADDRESS** | **DATA** |
| **A** | **2** |
| **B** | **3** |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **OUT PUT ADDRESS** | **DATA** |
| **R2** | **5** |
|  |  |

**SIMULATION**



**Program: 8-bit Subtraction:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Memory Location** | **Label** | **Opcode** | **Mnemonics** | **Comments** |
| 4100 |  | 74 05 | MOV A,#05 | Moves data 1 to register A |
| 4102 |  | 94 02 | SUBB A,#02 | Subtract data 2 from content of A and store result in A |
| 4104 |  | 90 45 00 | MOV DPTR,#4500 | Moves 4500 to DPTR |
| 4107 |  | F0 | MOVX @DPTR,A | Moves result by location by DTPR |
| 4108 |  | 80 FE | SJMP 4108 | Short jump to 4108 |

**Execution:**

**Subtraction:**

|  |  |
| --- | --- |
| ML | Input |
| 4101 |  |
| 4103 |  |

|  |  |
| --- | --- |
| ML | Output |
| 4500 |  |

**EDSIM51 PROGRAM-SUBTRACTION**

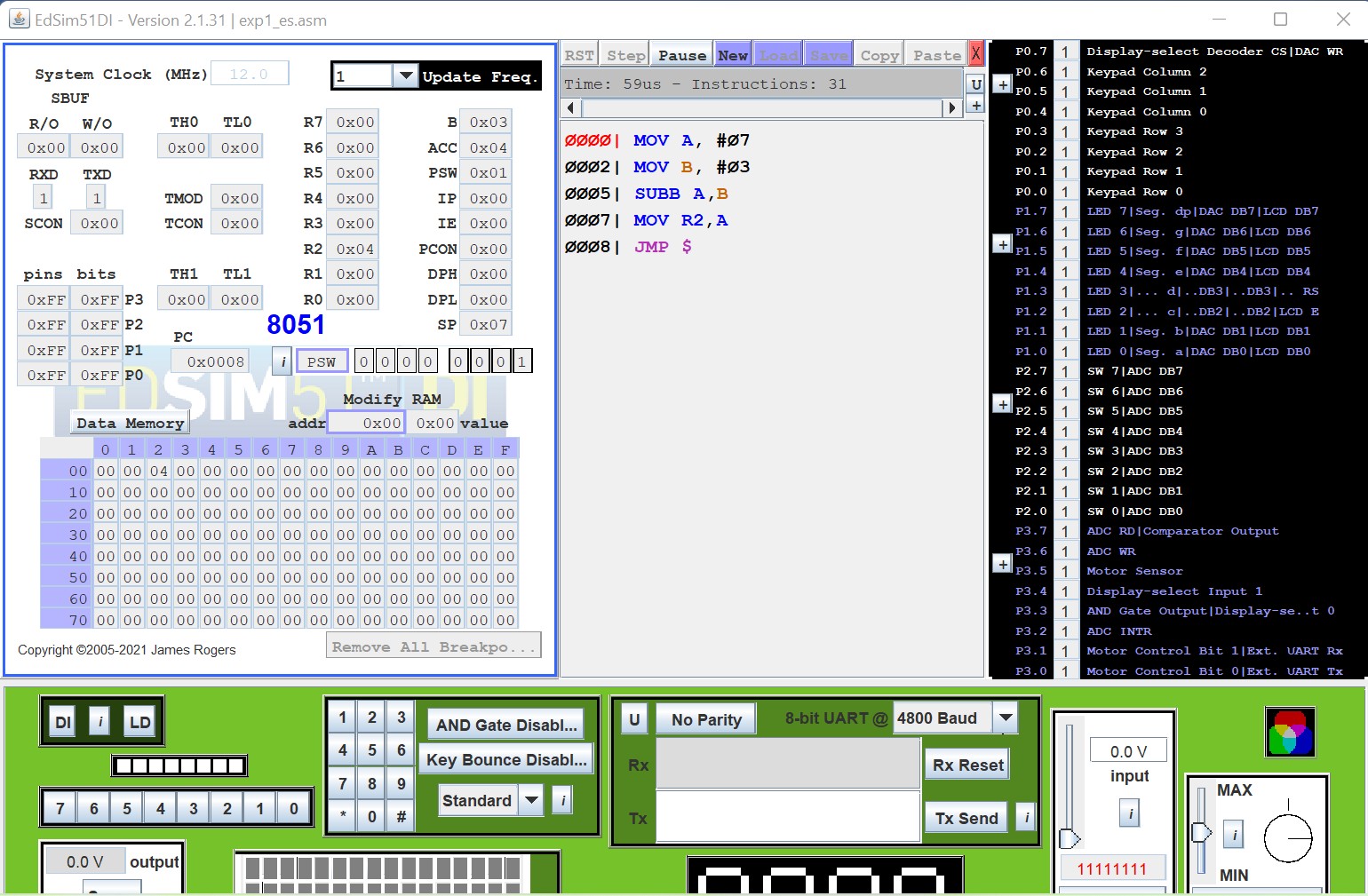
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ADDRESS** | **LABEL** | **MNEMONICS** | **OPCODE** | **COMMENTS** |
|  |  | MOV A, #Data |  | **Move a data (8 bit) to Acc** |
|  |  | MOV B, #Data |  | **Move a data (8 bit) to B Reg** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | SUBB A, B |  | **Subtract them, and store result in A** |
|  |  | MOV R0, A |  | **Copy the result to a Reg** |
|  |  | JMP $ |  | **Jump to Address location** |

|  |  |
| --- | --- |
| **IN PUT ADDRESS** | **DATA** |
| **A** | **7** |
| **B** | **3** |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **OUT PUT ADDRESS** | **DATA** |
| **R2** | **4** |
|  |  |

**SIMULATION**



**Program: 8-bit Multiplication:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Memory Location** | **Label** | **Opcode** | **Mnemonics** | **Comments** |
| 4100 | Start | 74 03 | MOV A,#03 | Move immediate data to accumulator |
| 4102 |  | 75 F0 02 | MOV B,#02 | Move 2nd data to B register |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4105 |  | A4 | MUL AB | Get the product in A & B |
| 4106 |  | 90 45 00 | MOV DPTR, # 4500 | Load data in 4500 location |
| 4109 |  | F0 | MOVX @DPTR,A | Move A t ext RAM |
| 410A |  | A3 | INC DPTR |  |
| 410B |  | E5 F0 | MOV A,B | Move 2nd data in A |
| 410D |  | F0 | MOVX @DPTR,A | Same the ext RAM |
| 410E |  | 80 FE | SJMP 410E | Remain idle in infinite loop |

**Execution: Multiplication:**

|  |  |
| --- | --- |
| ML | Input |
| 4101 |  |
| 4104 |  |

|  |  |
| --- | --- |
| Output Address | Value |
| 4500 |  |
| 4501 |  |

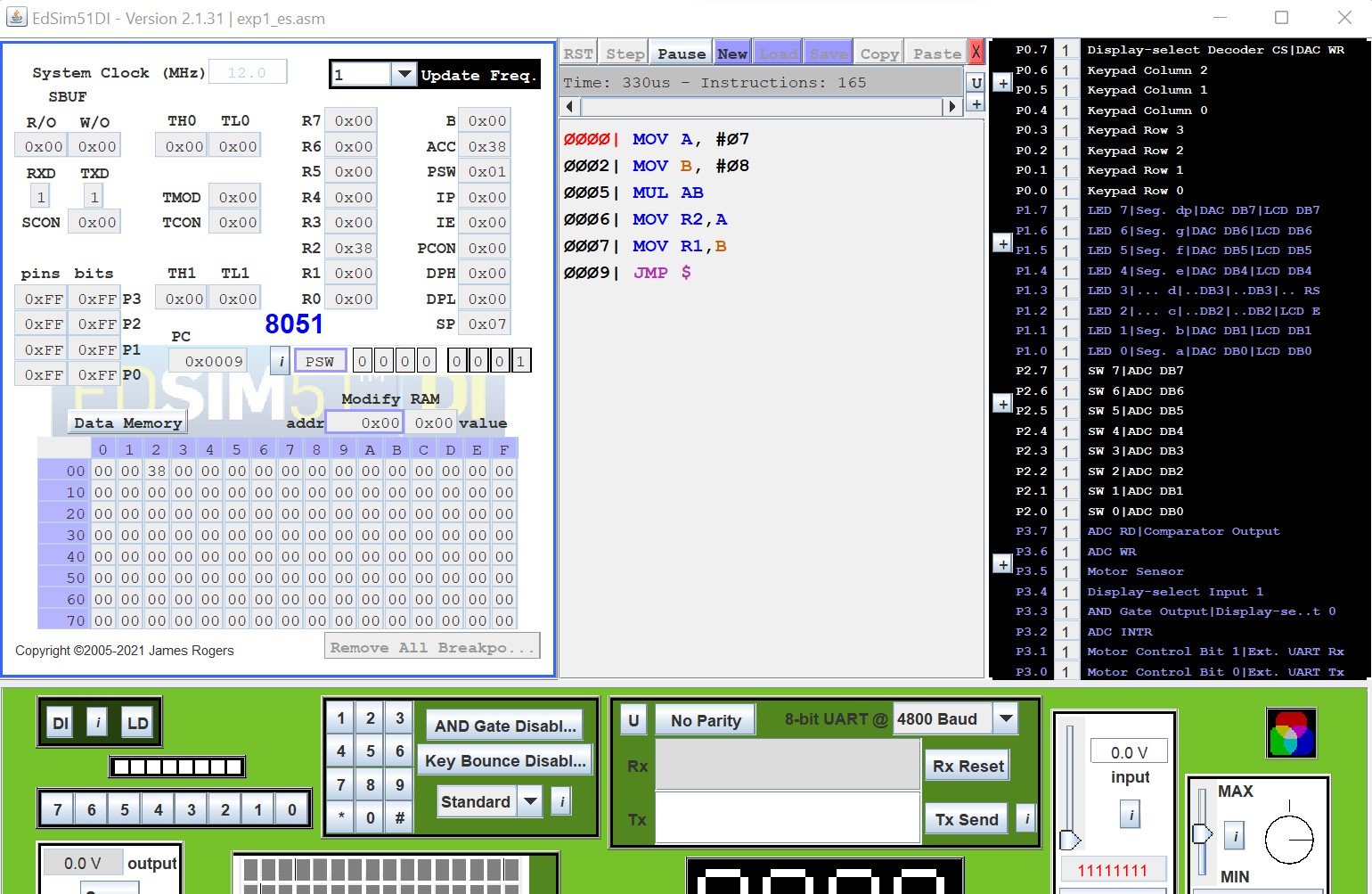
**EDSIM51 PROGRAM- MULTIPLICATION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ADDRESS** | **LABEL** | **MNEMONICS** | **OPCODE** | **COMMENTS** |
|  |  | MOV A, #Data |  | **Move a data (8 bit) to Acc** |
|  |  | MOV B, #Data |  | **Move a data (8 bit) to B Reg** |
|  |  | MUL AB |  | **Multiply them, and store result in A (and B- in case of 16 bit result)** |
|  |  | MOV R0, A |  | **Copy Lower half in A Reg** |
|  |  | MOV R1,B |  | **Copy Higher half in B Reg** |
|  |  | JMP $ |  | **Jump to Address location** |
|  |  |  |  |  |

|  |  |
| --- | --- |
| **IN PUT ADDRESS** | **DATA** |
| **A** | **07** |
| **B** | **08** |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **OUT PUT ADDRESS** | **DATA**  **(HEX)** |
| **R2** | **38** |
| **R1** | **00** |
|  |  |
|  |  |

**SIMULATION:**



**DIVISION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Memory Location** | **Label** | **Opcode** | **Mnemonics** | **Comments** |
| 4100 | Start | 74 04 | MOV A,#04 | Move immediate data to accumulator |
| 4102 |  | 75 F0 02 | MOV B,#02 | Move immediate to B  reg. |
| 4105 |  | 84 | DIV AB | Divide content of A & B |
| 4106 |  | 90 45 00 | MOV DPTR, # 4500 | Load data pointer with 4500 location |
| 4109 |  | F0 | MOVX @DPTR,A | Move A to ext RAM |
| 410A |  | A3 | INC DPTR | Increment data pointer |
| 410B |  | E5 F0 | MOV A,B | Move remainder to A |
| 410D |  | F0 | MOVX @DPTR,A | Move A to ext RAM |
| 410E |  | 80 FE | SJMP 410E | Remain idle in infinite loop |

**Execution: Division:**

|  |  |
| --- | --- |
| ML | Input |
| 4101 |  |
| 4103 |  |

|  |  |
| --- | --- |
| ML | Output |
| 4500 |  |
| 4501 |  |

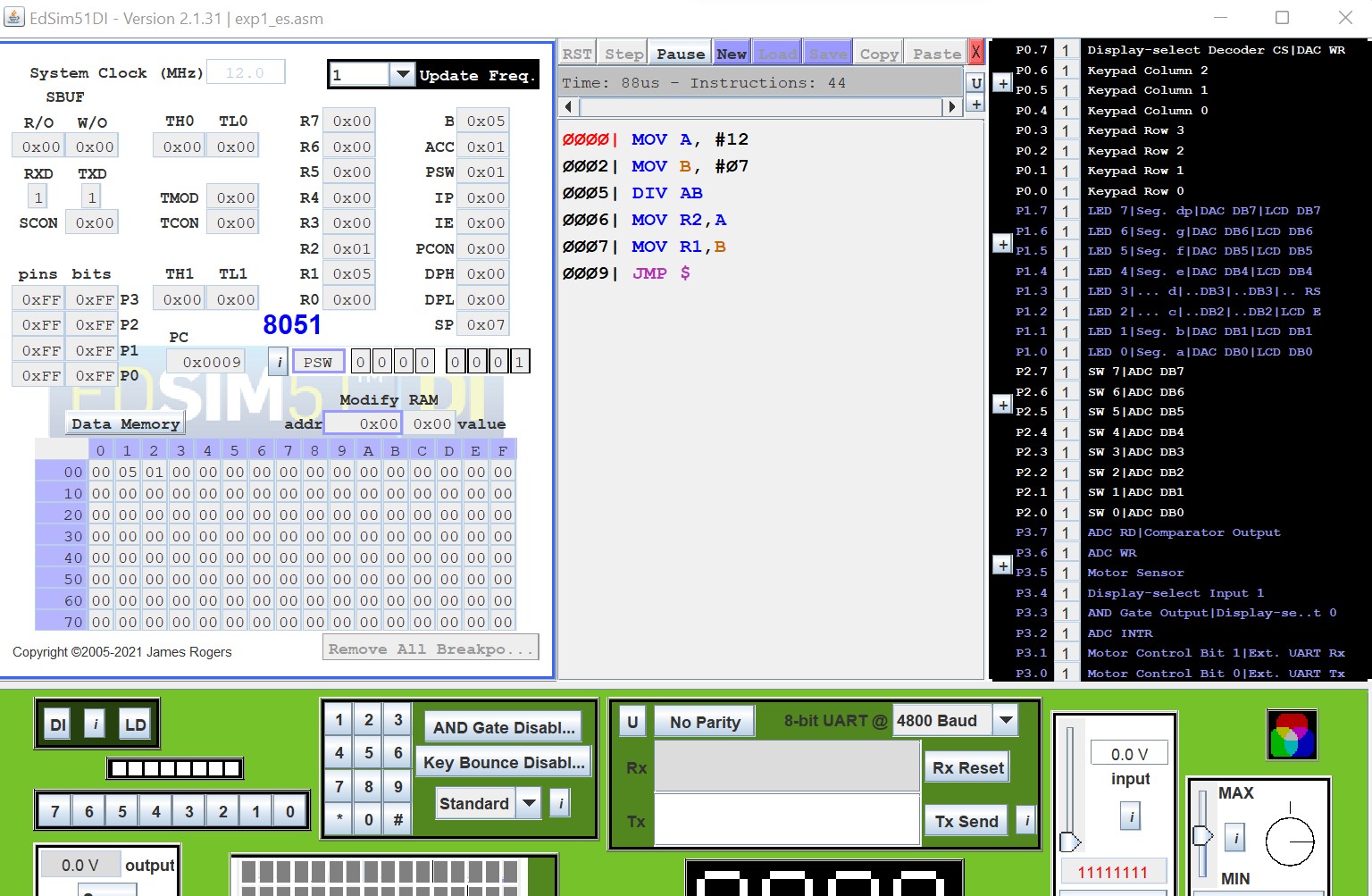
**EDSIM 51: DIVISION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ADDRESS** | **LABEL** | **MNEMONICS** | **OPCODE** | **COMMENTS** |
|  |  | MOV A, #Data |  | **Move a data (8 bit) to Acc** |
|  |  | MOV B, #Data |  | **Move a data (8 bit) to B Reg** |
|  |  | DIV AB |  | **Divide them, and store result in A (and B- in case of 16 bit result)** |
|  |  | MOV R0, A |  | **Copy Lower half in A Reg** |
|  |  | MOV R1, B |  | **Copy Higher half in B Reg** |
|  |  | JMP $ |  | **Jump to Address location** |

|  |  |
| --- | --- |
| **IN PUT ADDRESS** | **DATA** |
| **A** | **12** |
| **B** | **07** |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **OUT PUT ADDRESS** | **DATA** |
| **R2** | **01** |
| **R1** | **05** |
|  |  |
|  |  |

**SIMULATION:**



**Result:**

Thus 8-bit addition, subtraction, multiplication and division is performed using 8051

**POST-LAB**

###### Define OPCODE and Operand, and specify the opcode and the operand in the instruction MOV H, L.

Ans: The opcode is the instruction that is executed by the CPU and the operand is the data or memory location used to execute that instruction. MOV H, L, opcode is 74 L and operand is H.

###### Find the machine codes and the number of bytes of for the following instructions. Identify the opcode and the operands.

1. **MVI H,47H**
2. **ADI F5H**
3. **SUB C**

Ans)

1. MVI H,47 H

Machine code: 2647H

No. of bytes: 2byte instruction. Opcode =MVI

Operands=47H

1. ADIF5H

Machine code: C6F5H

No. of bytes: 2bytes instruction. Opcode=ADI

Operands= A(implied), F5H

1. SUB C

Machine code: 91C

No. of bytes: 1 byte instruction. Opcode=SUB Operands=A(implied), C

###### Find the HEX codes for the following instructions, identify the opcodes and operands, and show the order of entering the codes in memory

**STA 2050H JNZ 2070H**

Ans)

* + Hex Code of STA 2050H is 322050H where STA(STore Accumulator contents in memory) is opcode and 2050H is operand(address)
  + Hex Code of JNZ 2070H is C22070H where JNZ(Jump if Not Zero) is opcode and 2070H is operand(address)

###### Classification of 8051Instruction set.

Ans) Data transfer group – Example: MOV, MVI, LXI. Arithmetic group – Example: ADD, SUB, INR. Logical group – Example: ANA, XRA, CMP.

Branch group – Example: JMP, JNZ, CALL.

###### Find the hex machine code for the following instruction from the instruction set and identify the number of bytes of each instruction and assume that the starting address is 2000H.

**(MVI B,45H), (MVI C, 78H), (MOV A,C), ADD B OUT 07H**

**HLT.**

Ans)

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Hex code** | **bytes** | **address** |
| MVI B,4FH | 06 4F | 2 | 2000 |
| MVI C,78H | 0E 78 | 2 | 2002 |
| MOV A, C | 79 | 1 | 2004 |
| ADD B | 80 | 1 | 2005 |
| OUT 07H | D3 07 | 2 | 2006 |

|  |  |  |  |
| --- | --- | --- | --- |
| HLT | 76 | 1 | 2008 |

**DEPT. OF ELECTRICAL & ELECTRONICS ENGINEERING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, Kattankulathur – 603203.**

|  |
| --- |
| Title of Experiment : LED Blinking using 8051 Microcontroller |
| Name of the candidate : Rijo Saju Verghese  Register Number : RA1911044010016  Date of Experiment : 19/01/2022  Date of submission **: 30**/01/2022 |

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO:** | **MARKS SPLIT UP** | **MAXIMUM**  **MARKS (50)** | **MARKS**  **OBTAINED** |
| 1 | PRE LAB | 5 |  |
| 2 | PROGRAM | 25 |  |
| 3 | EXECUTION | 15 |  |
| 4 | POST LAB | 5 |  |
| TOTAL | | 50 |  |

**Staff Signature**

# LED Blinking using 8051 Microcontroller

#### **Aim:** To perform LED Blinking using 8051 microcontroller

**Apparatus required:**

1. **Hardware Requirement:**

8051 Microcontroller kit, Power supply

1. **Software Requirement:**

Keil uVision5

**Algorithm:**

Step 1: Declare integer i Step 2: Initialize P1 as 0

Step 3: Go to Step 4

Step 4: Initialize P1 as 0

Step 5: Initialize i as 0

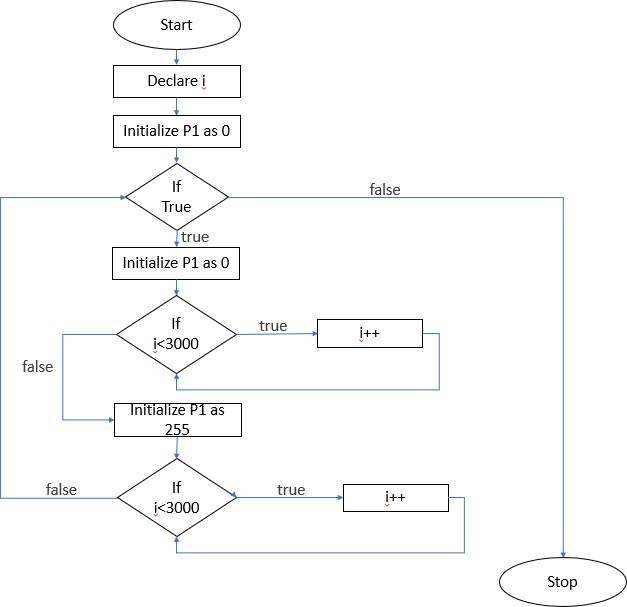
Step 6: If i is less than 3000, go to Step 7 else go to Step 8 Step 7: Increment i with 1, go to Step 6

Step 8: Initialize P1 as 255

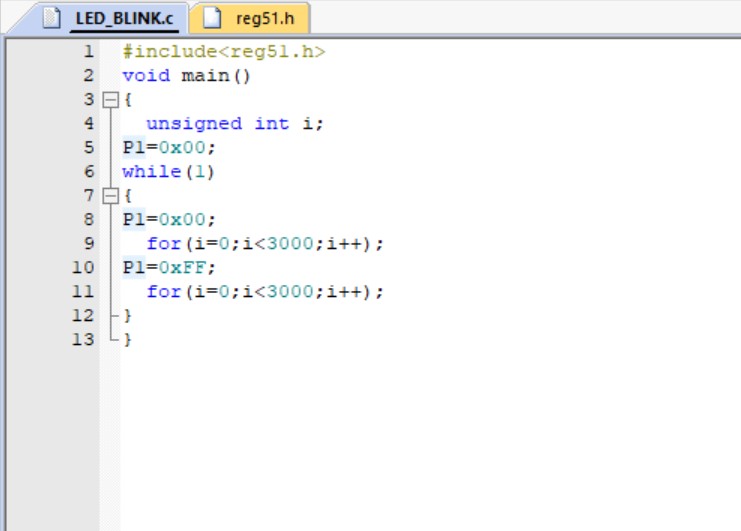
Step 9: Initialize i as 0

Step 10: If i is less than 3000, go to Step 11 else go to Step 12 Step 11: Increment i with 1, go to Step 10

Step 12: Go to Step 3

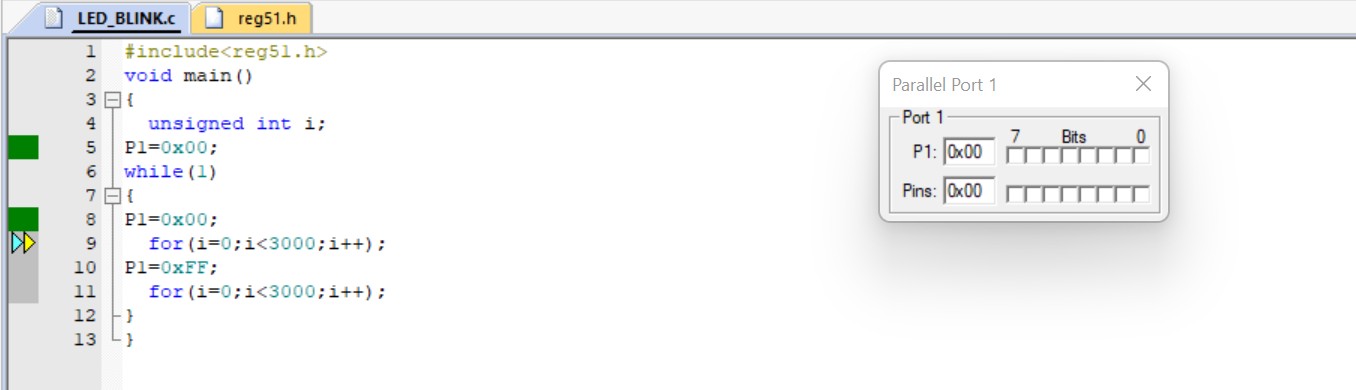


**Keil Program:**

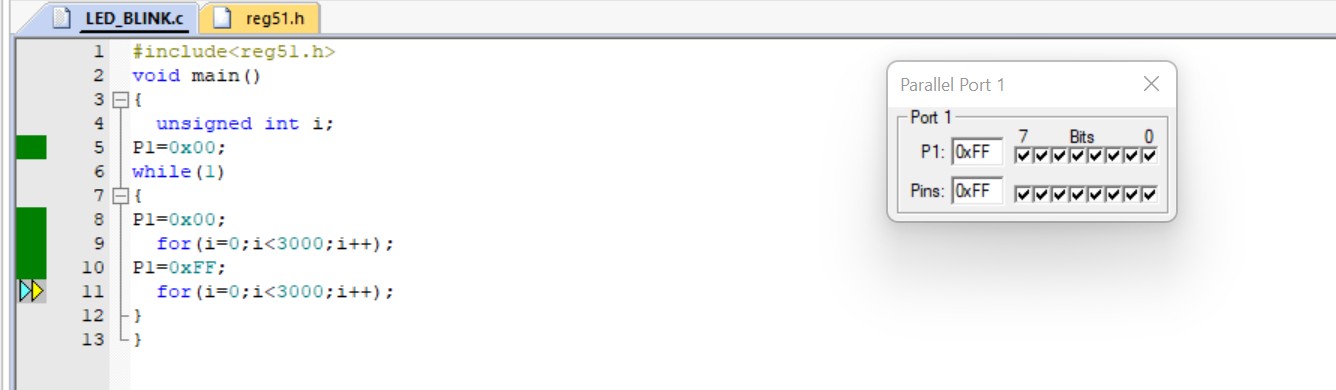


**Simulation:**

* + When P1 is set to 0



* + When P1 is set to 255



**Result:**

Thus, LED Blinking is performed using 8051 microcontroller.

**DEPT. OF ELECTRICAL & ELECTRONICS ENGINEERING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, Kattankulathur – 603203.**

|  |
| --- |
| Title of Experiment : PWM-Wave Generation |
| Name of the candidate : Rijo Saju Verghese  Register Number : RA1911044010016  Date of Experiment : 27/01/2022  Date of submission **: 03**/02/2022 |

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO:** | **MARKS SPLIT UP** | **MAXIMUM MARKS (50)** | **MARKS OBTAINED** |
| 1 | PRE LAB | 5 |  |
| 2 | PROGRAM | 25 |  |
| 3 | EXECUTION | 15 |  |
| 4 | POST LAB | 5 |  |
| TOTAL | | 50 |  |

**Staff Signature**

# PWM-Wave Generation

**Aim:** To perform the Pulse Width Modulation (PWM)-Wave Generation using 8051 Microcontroller

### Apparatus required:

**Hardware Requirement:**

8051 Microcontroller kit, Power supply

### Software Requirement:

Keil uVision5

### Calculation:

The frequency for the timer is always 1/12th of the frequency of the crystal attached to the 8051.

Thus, TF = 1/12 x 11.059MHz

= 921583 Hz

Approx. TF = 922kHz Timer’s clock period:

TP = 1/TF

TP = 1/ 922= 1.085 m sec

Therefore, if we multiple this by 922 we get TP = 1ms

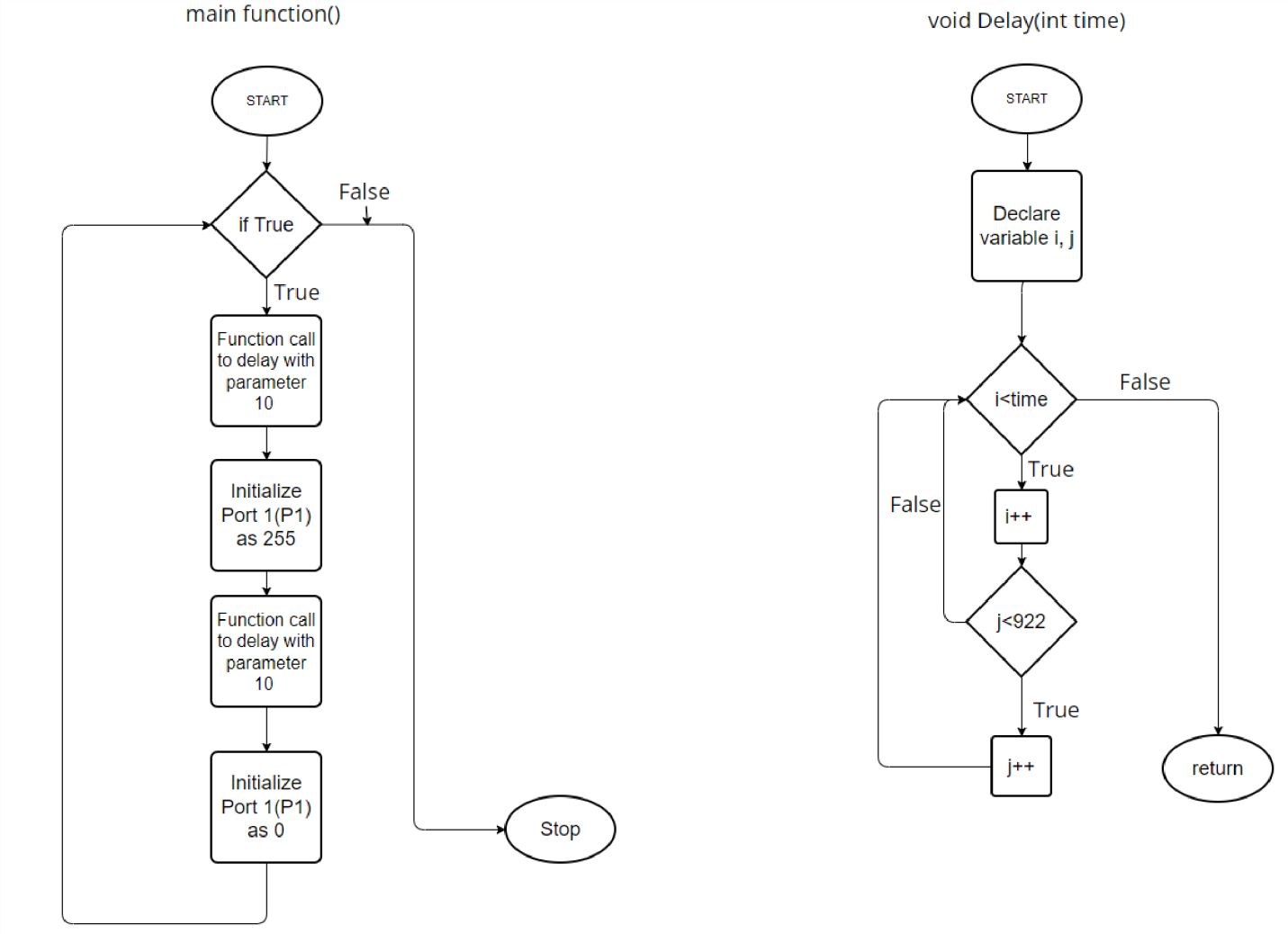
### Algorithm:

Step 1: If false, STOP else Go to Step 2 Step 2: call delay function with parameter 10 Step 3: Initialize P1 as 255

Step 4: call delay function with parameter 10 Step 5: Initialize P1 as 0

Step 6: Go to Step 1

### Flowchart:



**Keil Program:**

#include<reg51.h> void delay(int time)

{

int i,j; for(i=0;i<time;i++) for(j=0;j<922;j++);

}

void main()

{

while(1)

{

P1=255;

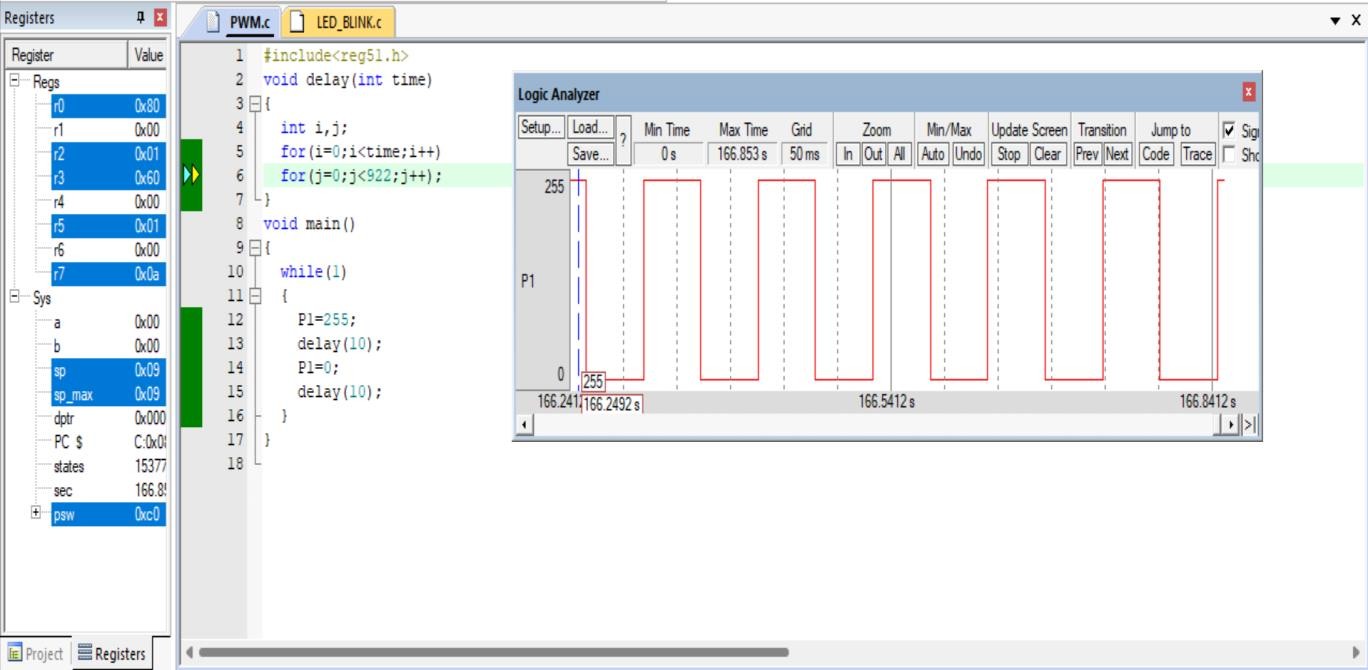
delay(10); P1=0;

delay(10);

}

}

### Simulation:



**Result:** Thus, Pulse Width Modulation (PWM)-Wave Generation is performed using 8051 microcontroller.

**DEPT. OF ELECTRICAL & ELECTRONICS ENGINEERING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, Kattankulathur – 603203.**

|  |
| --- |
| Title of Experiment : Line Follower Robot |
| Name of the candidate :Rijo Saju Verghese  Register Number :RA1911044010016  Date of Experiment :04-02-2022  Date of submission **:10-02-2022** |

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO**  **:** | **MARKS SPLIT UP** | **MAXIMUM MARKS (50)** | **MARKS OBTAINED** |
| 1 | PRE LAB | 5 |  |
| 2 | PROGRAM | 25 |  |
| 3 | EXECUTION | 15 |  |
| 4 | POST LAB | 5 |  |
| TOTAL | | 50 |  |

**Staff Signature**

##### Aim:

To construct the Line Follower Robot Using Arduino.

##### Apparatus required:

* Hardware Requirement :

Arduino, IR Sensor, DC Motor, Robot, Motor Driver

##### Software Requirement :

Arduino software

##### Source Code:

int vSpeed = 110; // MAX 255 int turn\_speed = 230; // MAX 255 int turn\_delay = 10;

//L293 Connection

const int motorA1 = 8; const int motorA2 = 10; const int motorAspeed = 9; const int motorB1 = 12; const int motorB2 = 13; const int motorBspeed = 11;

//Sensor Connection

const int left\_sensor\_pin =A0; const int right\_sensor\_pin =A1;

int left\_sensor\_state; int right\_sensor\_state;

void setup() { pinMode(motorA1, OUTPUT); pinMode(motorA2, OUTPUT); pinMode(motorB1, OUTPUT); pinMode(motorB2, OUTPUT);

Serial.begin(9600); delay(3000);

}

void loop() {

left\_sensor\_state = analogRead(left\_sensor\_pin); right\_sensor\_state = analogRead(right\_sensor\_pin);

if(right\_sensor\_state > 500 && left\_sensor\_state < 500)

{

Serial.println("turning right");

digitalWrite (motorA1,LOW); digitalWrite(motorA2,HIGH); digitalWrite (motorB1,LOW); digitalWrite(motorB2,HIGH);

analogWrite (motorAspeed, vSpeed); analogWrite (motorBspeed, turn\_speed);

}

if(right\_sensor\_state < 500 && left\_sensor\_state > 500)

{

Serial.println("turning left");

digitalWrite (motorA1,HIGH); digitalWrite(motorA2,LOW); digitalWrite (motorB1,HIGH); digitalWrite(motorB2,LOW);

analogWrite (motorAspeed, turn\_speed); analogWrite (motorBspeed, vSpeed);

delay(turn\_delay);

}

if(right\_sensor\_state > 500 && left\_sensor\_state > 500)

{

Serial.println("going forward");

digitalWrite (motorA2,LOW); digitalWrite(motorA1,HIGH); digitalWrite (motorB2,HIGH); digitalWrite(motorB1,LOW); analogWrite (motorAspeed, vSpeed); analogWrite (motorBspeed, vSpeed); delay(turn\_delay);

}

## if(right\_sensor\_state < 500 && left\_sensor\_state < 500)

{

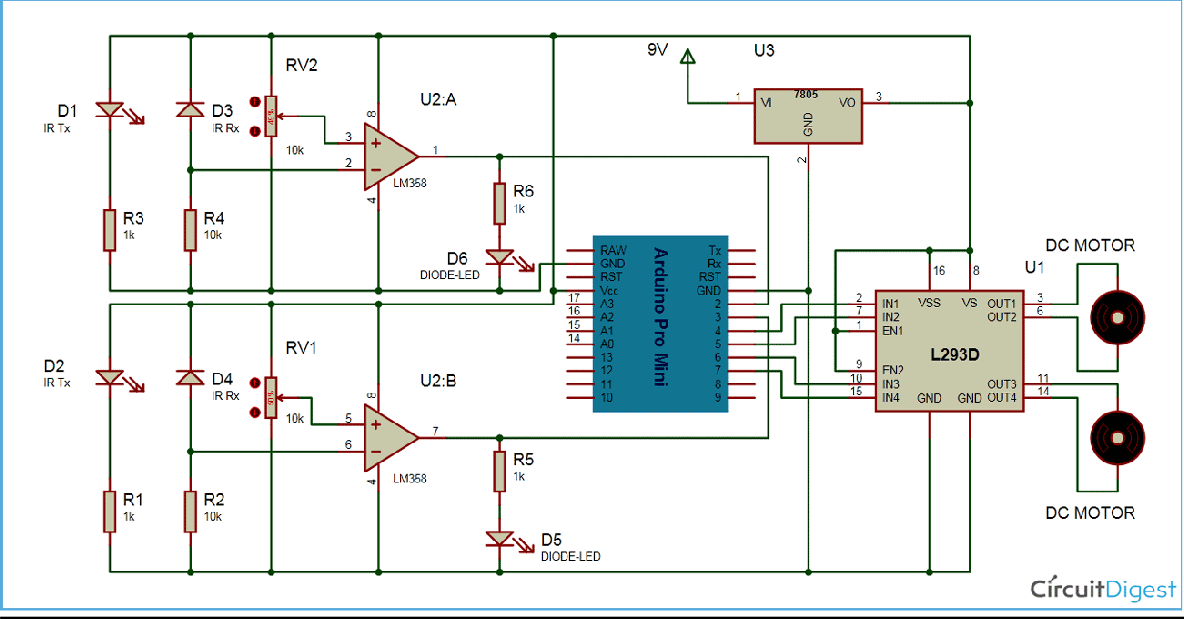
Serial.println("stop"); analogWrite (motorAspeed, 0);

analogWrite (motorBspeed, 0);

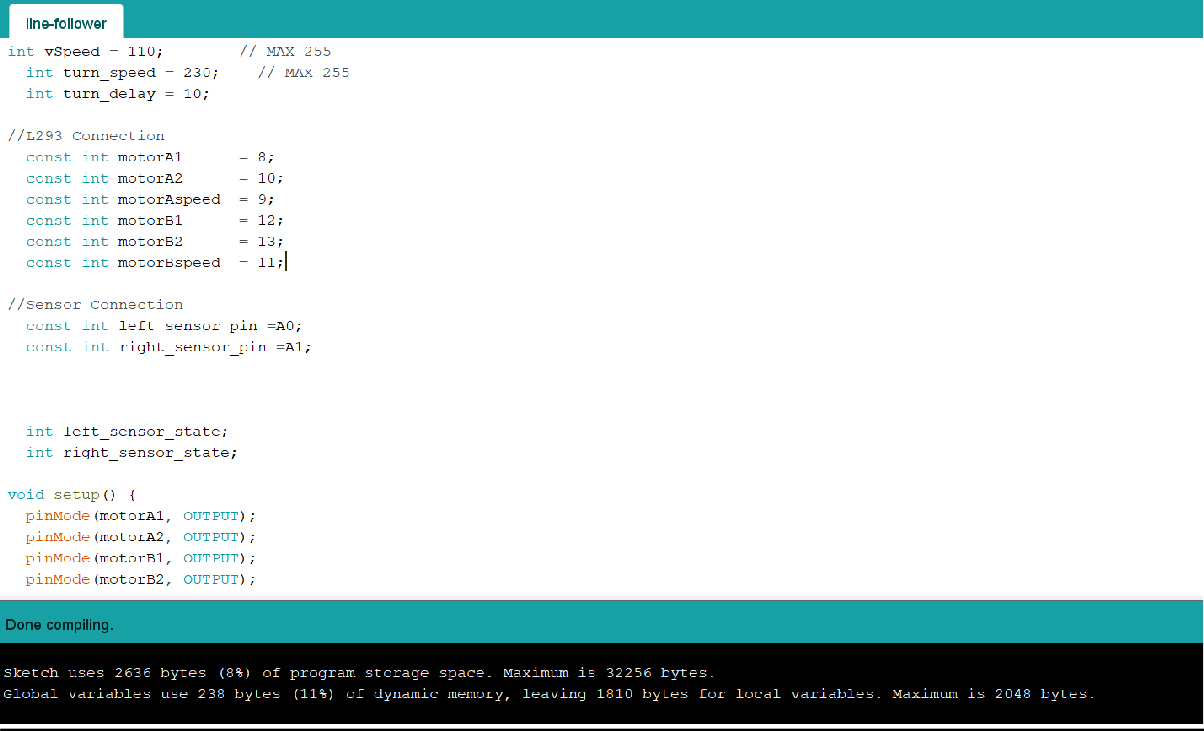
}

}

**CIRCUIT:**

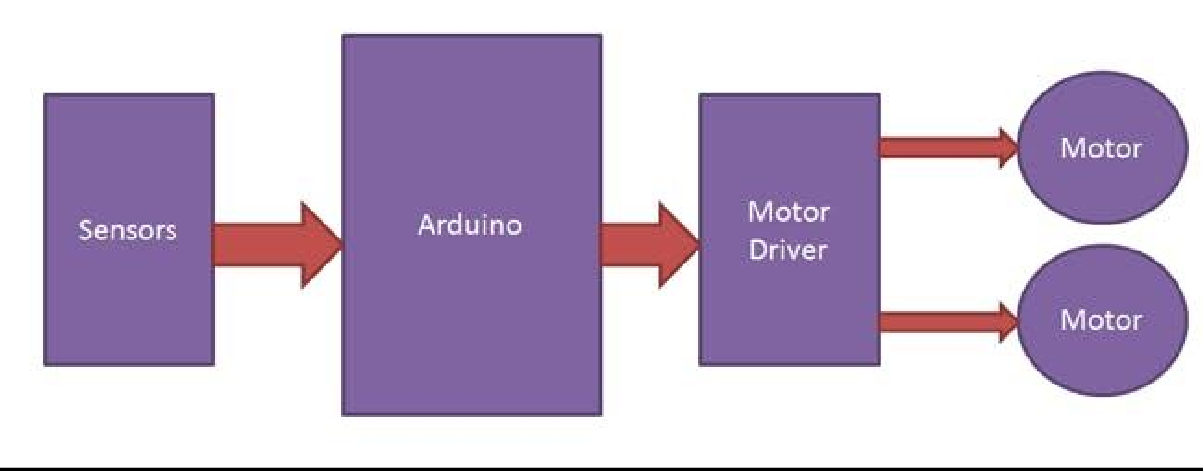


**SIMULATION**

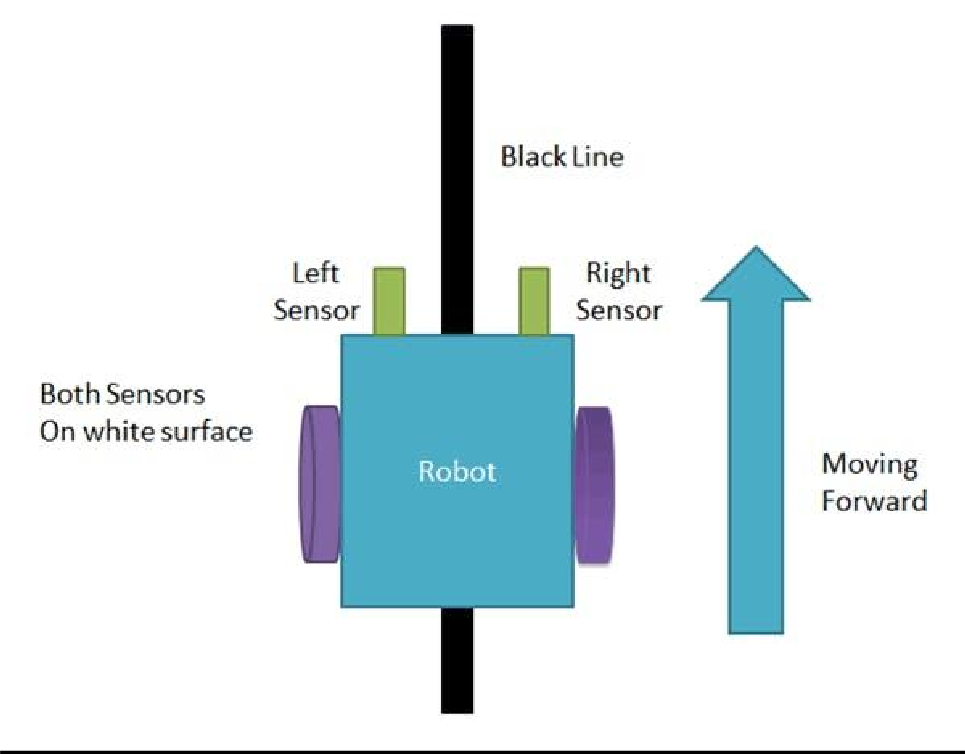


***Working Of Line Follower Robot***

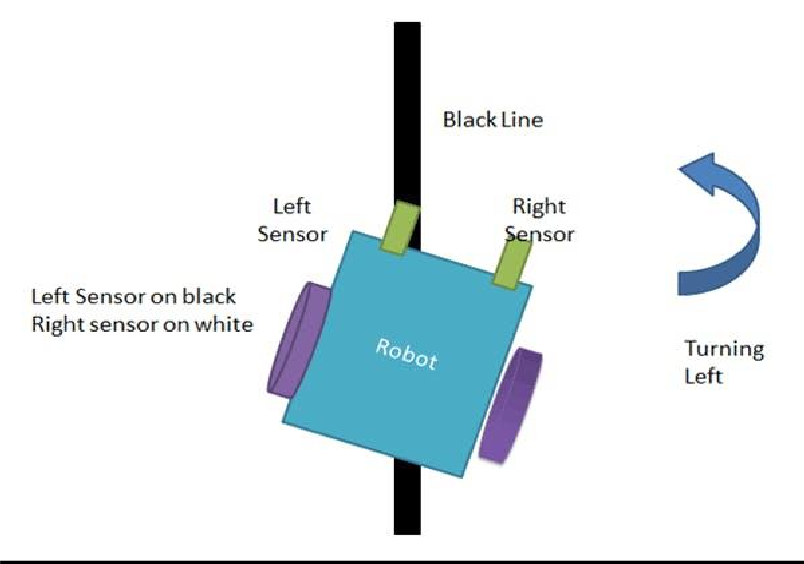
*Line follower robot senses black line by using sensor and then sends the signal to arduino. Then arduino drives the motor according to sensors' output.*



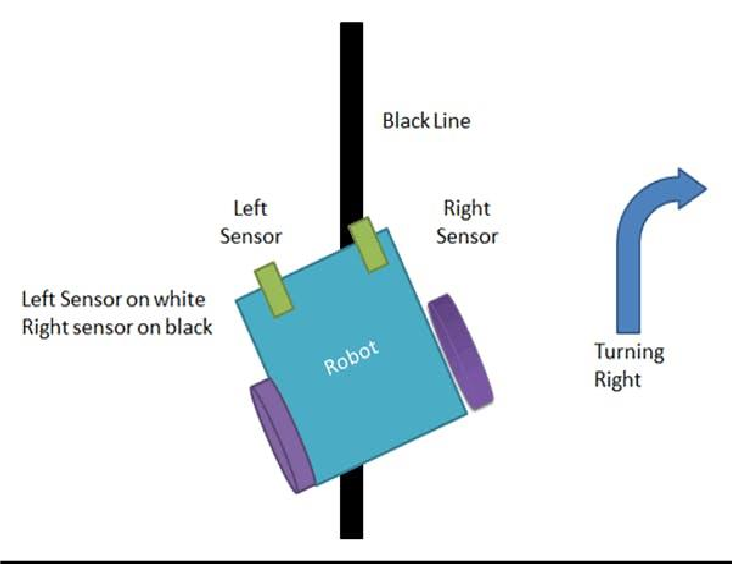
*Here in this project, we are using two IR sensor modules namely left sensor and right sensor. When both left and right sensor senses white then robot move forward.*



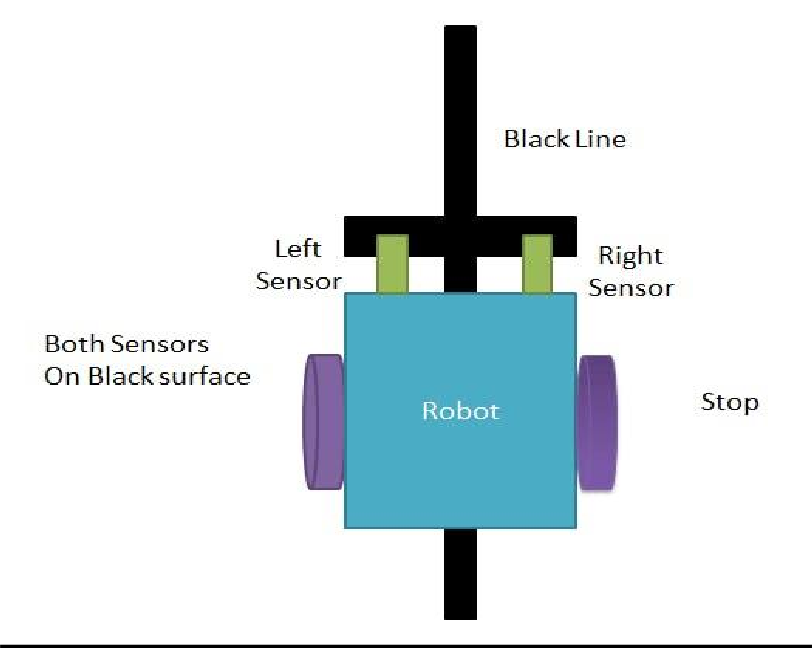
*If left sensor comes on black line then robot turn left side.*



*If right sensor sense black line then robot turn right side until both sensor comes at white surface. When white surface comes robot starts moving on forward again.*



*If both sensors comes on black line, robot stops.*



***Result:***

*Thus the project Line Follower Robot is performed using Arduino*

**DEPT. OF ELECTRICAL & ELECTRONICS ENGINEERING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, Kattankulathur – 603203.**

|  |
| --- |
| Title of Experiment : Arduino based Sun Tracking Solar Panel |
| Name of the candidate :Rijo Saju Verghese  Register Number :RA1911044010016  Date of Experiment :11-02-2022  Date of submission **:15-02-2022** |

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO**  **:** | **MARKS SPLIT UP** | **MAXIMUM MARKS (50)** | **MARKS OBTAINED** |
| 1 | PRE LAB | 5 |  |
| 2 | PROGRAM | 25 |  |
| 3 | EXECUTION | 15 |  |
| 4 | POST LAB | 5 |  |
| TOTAL | | 50 |  |

**Staff Signature**

##### Aim:

To construct an Arduino based Sun Tracking Solar Panel

##### Apparatus required:

Hardware Requirement:

1. 5mm Acrylic Sheet 30cm x 20cm,
2. VPC Card board 5mm,
3. Arduino UNO,
4. LDR Sensor x 2,
5. Mini Servo Motor SG90,
6. USB DC 1~5V to DC 5V Voltage Step Up Boost Module,
7. LiPo Battery Charger Module Mini TP4056 IC,
8. On/Off Switch,
9. 18650 Battery Holder – 1 Cell,
10. 18650 Battery Cell 3.7V

##### Software Requirement:

Arduino software

##### Source Code:

#include <Servo.h> //includes the servo library Servo myservo;

#define ldr1 A0 // set ldr 1 Analog input pin of East ldr as an integer #define ldr2 A1 // set ldr 2 Analog input pin of West ldr as an integer

int pos = 90; // initial position of the Horizontal movement controlling servo motor

int tolerance = 20; // allowable tolerance setting - so solar servo motor isn't constantly in motion

void setup(){

myservo.attach(2); // attaches the servo on digital pin 2 to the horizontal movement servo motor

pinMode(ldr1, INPUT); //set East ldr pin as an input pinMode(ldr2, INPUT); //set West ldr pin as an input

myservo.write(pos); // write the starting position of the horizontal movement servo motor

delay(1000); // 1 second delay to allow the solar panel to move to its staring position before comencing solar tracking

}

void loop(){

int val1 = analogRead(ldr1); // read the value of ldr 1 int val2 = analogRead(ldr2); // read the value of ldr 2

if((abs(val1 - val2) <= tolerance) || (abs(val2 - val1) <= tolerance)) {

//no servo motor horizontal movement will take place if the ldr value is within the allowable tolerance

}else {

if(val1 > val2) // if ldr1 senses more light than ldr2

{

pos = pos+1; // decrement the 90 degree poistion of the horizontal servo motor - this will move the panel position Eastward

}

if(val1 < val2) // if ldr2 senses more light than ldr1

{

pos = pos-1; // increment the 90 degree position of the horizontal motor - this will move the panel position Westward

}

}

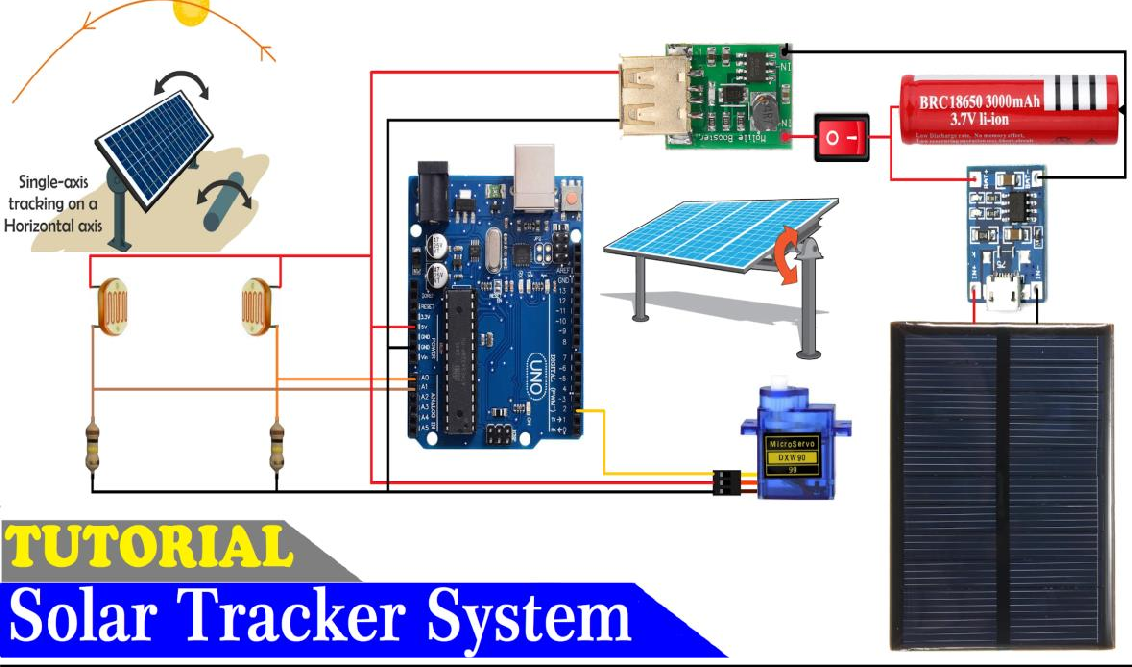
if(pos > 180) {pos = 180;} // reset the horizontal postion of the motor to 180 if it tries to move past this point

if(pos < 0) {pos = 0;} // reset the horizontal position of the motor to 0 if it tries to move past this point

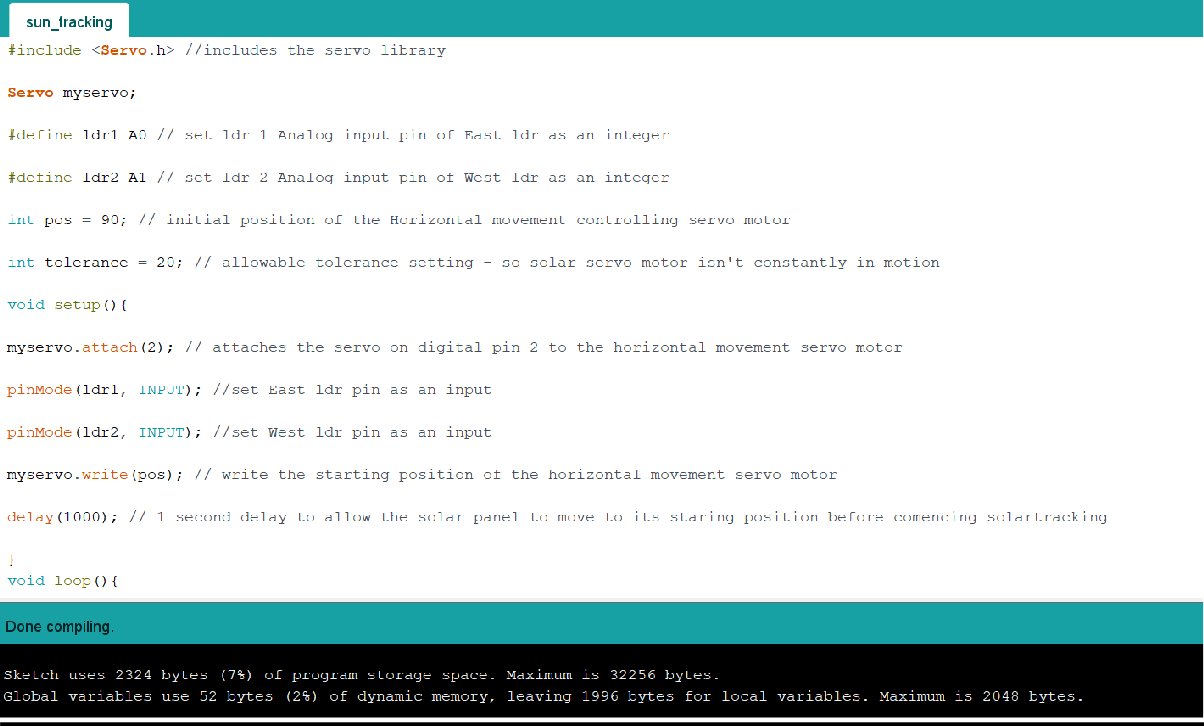
myservo.write(pos); // write the starting position to the horizontal motor delay(50);

}

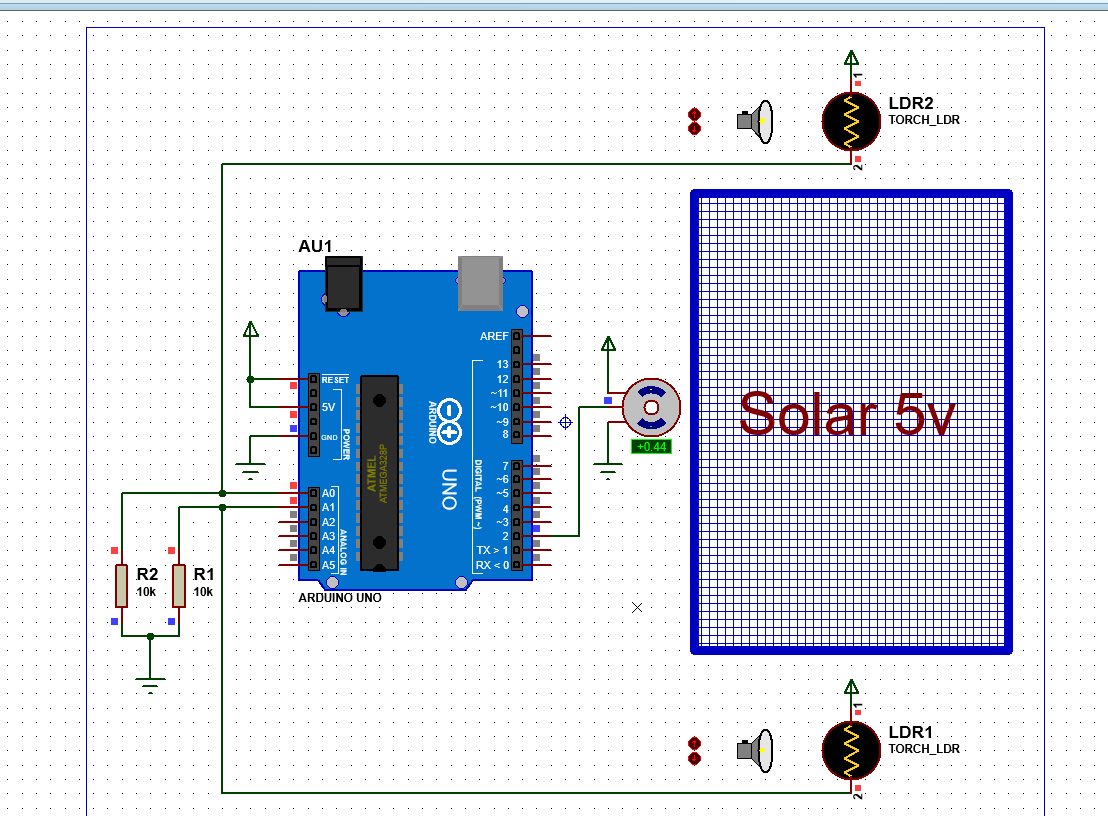
**SOLAR TRACKING SYSTEM**



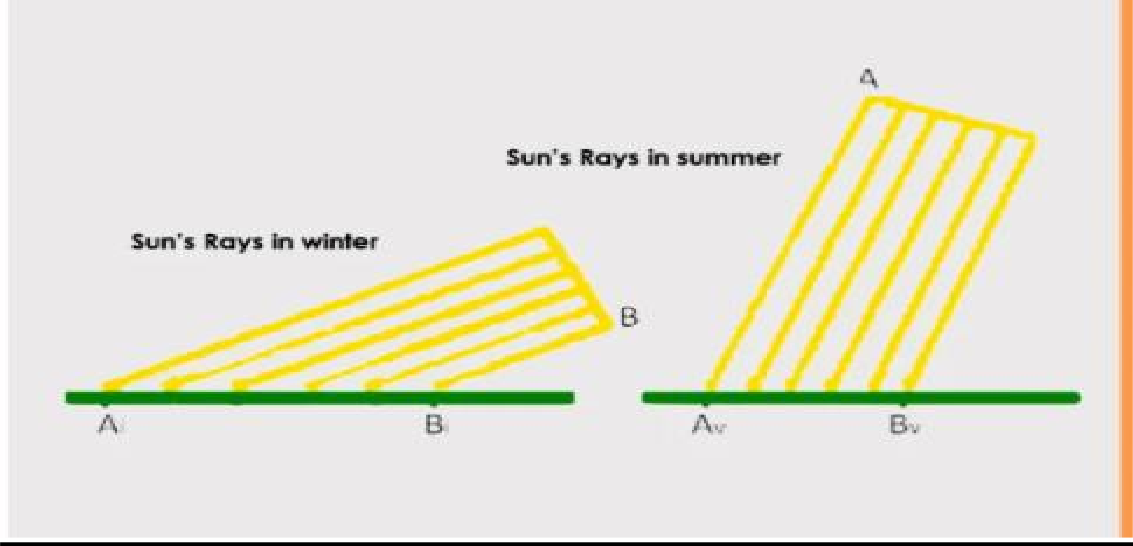
**SIMULATION**



***Proteus Simulation***



***Position Of The Sun***



***Result:***

*Thus, Arduino based Sun Tracking Solar Panel is constructed using Arduino*

**DEPT. OF ELECTRICAL & ELECTRONICS ENGINEERING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, Kattankulathur – 603203.**

|  |
| --- |
| Title of Experiment : Traffic Light Control using ATMega250 |
| Name of the candidate : Rijo Saju Verghese  Register Number : RA1911044010016  Date of Experiment :26-02-2022  Date of submission **:28-02-2022** |

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO**  **:** | **MARKS SPLIT UP** | **MAXIMUM MARKS (50)** | **MARKS OBTAINED** |
| 1 | PRE LAB | 5 |  |
| 2 | PROGRAM | 25 |  |
| 3 | EXECUTION | 15 |  |
| 4 | POST LAB | 5 |  |
| TOTAL | | 50 |  |

**Staff Signature**

##### Aim:

To peform Traffic Light Control Using ATMega250.

##### Apparatus required:

Hardware Requirement :

1. Arduino Mega
2. *LED’s (red, yellow, green)*
3. *connecting wires*
4. *Beard board*
5. *Resistors*

***Software Requirement :***

Arduino software

##### Source Code:

int signal1[] = {23, 25, 27};

int signal2[] = {46, 48, 50};

int signal3[] = {13, 12, 11};

int signal4[] = {10, 9, 8}; int redDelay = 5000;

int yellowDelay = 2000; void setup() {

// Declaring all the LED's as output for (int i = 0; i < 3; i++) { pinMode(signal1[i], OUTPUT); pinMode(signal2[i], OUTPUT); pinMode(signal3[i], OUTPUT); pinMode(signal4[i], OUTPUT);

}

}

void loop() {

// Making Green LED at signal 1 and red LED's at other signal HIGH digitalWrite(signal1[2], HIGH);

digitalWrite(signal1[0], LOW); digitalWrite(signal2[0], HIGH);

digitalWrite(signal3[0], HIGH); digitalWrite(signal4[0], HIGH); delay(redDelay);

// Making Green LED at signal 1 LOW and making yellow LED at signal 1 HIGH for 2 seconds

digitalWrite(signal1[1], HIGH); digitalWrite(signal1[2], LOW); delay(yellowDelay); digitalWrite(signal1[1], LOW);

// Making Green LED at signal 2 and red LED's at other signal HIGH digitalWrite(signal1[0], HIGH);

digitalWrite(signal2[2], HIGH); digitalWrite(signal2[0], LOW); digitalWrite(signal3[0], HIGH); digitalWrite(signal4[0], HIGH); delay(redDelay);

// Making Green LED at signal 2 LOW and making yellow LED at signal 2 HIGH for 2 seconds

digitalWrite(signal2[1], HIGH); digitalWrite(signal2[2], LOW); delay(yellowDelay); digitalWrite(signal2[1], LOW);

// Making Green LED at signal 3 and red LED's at other signal HIGH digitalWrite(signal1[0], HIGH);

digitalWrite(signal2[0], HIGH); digitalWrite(signal3[2], HIGH); digitalWrite(signal3[0], LOW); digitalWrite(signal4[0], HIGH); delay(redDelay);

// Making Green LED at signal 3 LOW and making yellow LED at signal 3 HIGH for 2 seconds

digitalWrite(signal3[1], HIGH); digitalWrite(signal3[2], LOW); delay(yellowDelay); digitalWrite(signal3[1], LOW);

// Making Green LED at signal 4 and red LED's at other signal HIGH digitalWrite(signal1[0], HIGH);

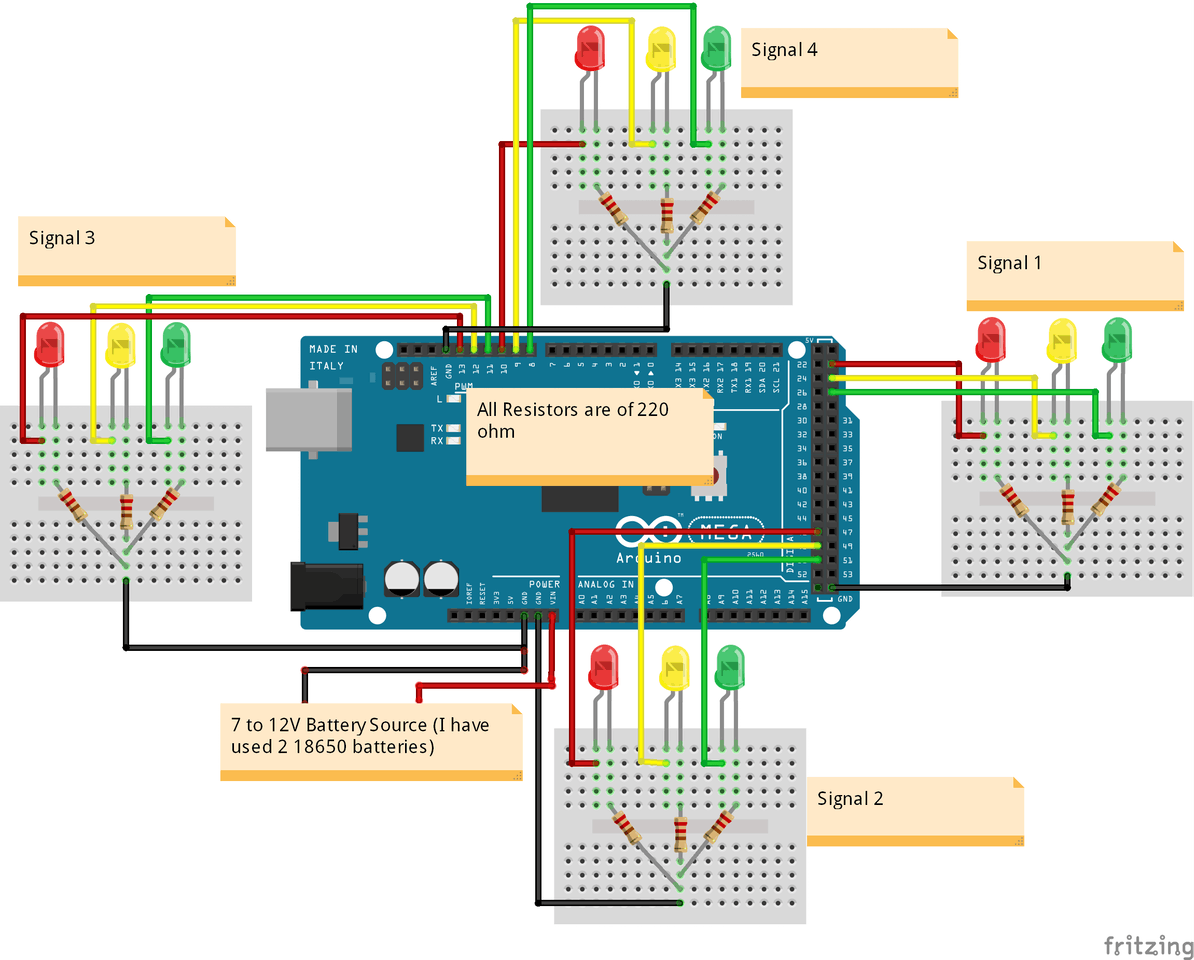
digitalWrite(signal2[0], HIGH); digitalWrite(signal3[0], HIGH); digitalWrite(signal4[2], HIGH); digitalWrite(signal4[0], LOW); delay(redDelay);

// Making Green LED at signal 4 LOW and making yellow LED at signal 4 HIGH for 2 seconds

digitalWrite(signal4[1], HIGH); digitalWrite(signal4[2], LOW); delay(yellowDelay); digitalWrite(signal4[1], LOW);

}

**CIRCUIT:**



**SIMULATION**



***Result:***

*Thus,* Traffic Light Control *is performed using ATMega250.*

**DEPT. OF ELECTRICAL & ELECTRONICS ENGINEERING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, Kattankulathur – 603203.**

|  |
| --- |
| Title of Experiment : DC Motor Speed Control Using Arduino |
| Name of the candidate : Rijo Saju Verghese  Register Number :RA1911044010016  Date of Experiment :03-03-2022  Date of submission **:10-03-2022** |

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO**  **:** | **MARKS SPLIT UP** | **MAXIMUM MARKS (50)** | **MARKS OBTAINED** |
| 1 | PRE LAB | 5 |  |
| 2 | PROGRAM | 25 |  |
| 3 | EXECUTION | 15 |  |
| 4 | POST LAB | 5 |  |
| TOTAL | | 50 |  |

**Staff Signature**

##### Aim:

To control/regulate the DC motor using Arduino and Potentiometer.

##### Apparatus required:

Hardware Requirement:

1. Arduino UNO
2. DC motor (generic)
3. Jumper wires (generic)
4. Breadboard (generic)
5. 9V battery (generic)
6. Single Turn Potentiometer- 10k ohms
7. SparkFun Full-Bridge Motor Driver Breakout - L298N
8. USB-A to B Cable

##### Software Requirement:

Arduino software

##### Source Code:

void setup()

{

Serial.begin(9600); pinMode(3,OUTPUT); pinMode(4,OUTPUT); digitalWrite(4,LOW); pinMode(A0,INPUT);

}

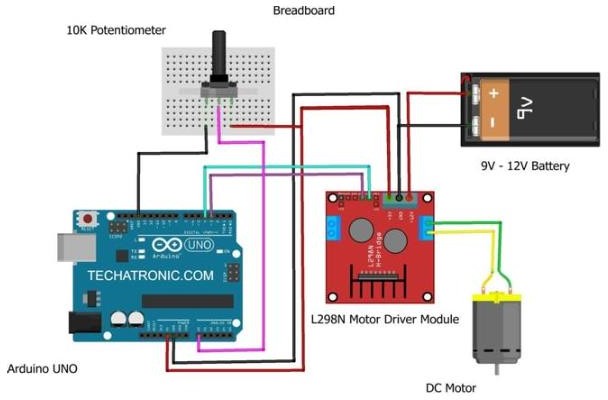
void loop()

{

int s=analogRead(A0);

int z=map(s,0,1024,0,255); Serial.println(z); analogWrite(3,z);

}

**CIRCUIT:**

**SIMULATION**



**)**

***Result:***

*Thus,* DC Motor is controlled using *Arduino and Potentiometer.*

**DEPT. OF ELECTRICAL & ELECTRONICS ENGINEERING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, Kattankulathur – 603203.**

|  |
| --- |
| Title of Experiment : Automatic water level indicator and Controller using Arduino |
| Name of the candidate : Rijo Saju Verghese  Register Number : RA1911044010016  Date of Experiment :10-03-2022  Date of submission **:17-03-2022** |

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO**  **:** | **MARKS SPLIT UP** | **MAXIMUM**  **MARKS (50)** | **MARKS**  **OBTAINED** |
| 1 | PRE LAB | 5 |  |
| 2 | PROGRAM | 25 |  |
| 3 | EXECUTION | 15 |  |
| 4 | POST LAB | 5 |  |
| TOTAL | | 50 |  |

**Staff Signature**

##### Aim:

To construct Automatic water level indicator and Controller using Arduino.

##### Apparatus required:

Hardware Requirement:

1. Arduino UNO
2. Ultrasonic sensor Module
3. Adafruit RGB Backlight LCD - 16x2
4. Relay 6 Volt
5. Texas Instruments BOOSTXL-ULN2003 ULN2003A Dual Stepper MotorBoosterPack
6. 9 volt battery
7. Connecting wires

##### Software Requirement:

Arduino software

##### Source Code:

#include <LiquidCrystal.h> #define trigger 10

#define echo 11

#define motor 8

#define buzzer 12

LiquidCrystal lcd(7, 6, 5, 4, 3, 2); float time = 0, distance = 0;

void setup()

{

lcd.begin(16, 2); pinMode(trigger, OUTPUT); pinMode(echo, INPUT); pinMode(motor, OUTPUT); pinMode(buzzer, OUTPUT); lcd.print(" Water Level "); lcd.setCursor(0, 1); lcd.print(" Indicator "); delay(2000);

}

void loop()

{

lcd.clear(); digitalWrite(trigger, LOW); delayMicroseconds(2); digitalWrite(trigger, HIGH); delayMicroseconds(10); digitalWrite(trigger, LOW); delayMicroseconds(2);

time = pulseIn(echo, HIGH); distance = time \* 340 / 20000; lcd.clear();

lcd.print("Water Space In "); lcd.setCursor(0, 1); lcd.print("Tank is: "); lcd.print(distance); lcd.print("Cm"); delay(2000);

if (distance < 12)

{

digitalWrite(motor, LOW); digitalWrite(buzzer, HIGH); lcd.clear();

lcd.print("Water Tank Full "); lcd.setCursor(0, 1); lcd.print("Motor Turned OFF"); delay(2000); digitalWrite(buzzer, LOW); delay(3000);

}

else if (distance > 30)

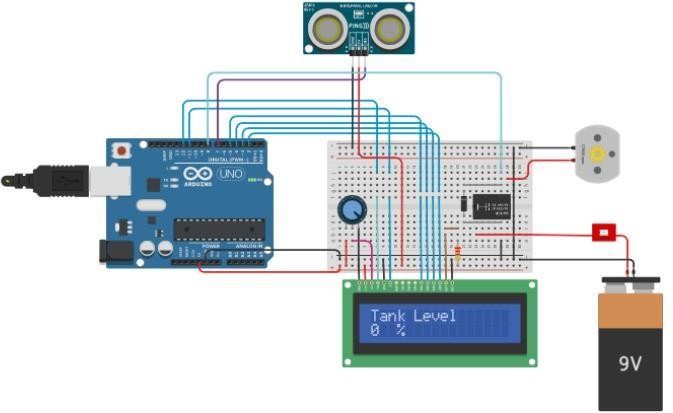
{

digitalWrite(motor, HIGH); lcd.clear();

lcd.print("LOW Water Level"); lcd.setCursor(0, 1); lcd.print("Motor Turned ON"); delay(5000);

}

}

**CIRCUIT:**

**SIMULATION**



***Result:***

*Thus,* Water Level is controlled/indicated using *Arduino.*