

Reg no: 25RP22821

Department: ADVANCED DIPLOMA in EIT

Y1 EITB

Module code: EITMM601: Microprocessor and Microcontroller

1. Given data RAM of 4KB, with intel 404 microprocessor to find the number of address lines needed to connect a 4KB RAM:

$$4KB = 4096 \text{ bytes}$$

2^n is memory capacity and n is number of address line

$$4096 = 2^{12}$$

$$2^n = 2^{12} \quad n = 12$$

Addressing 4KB directly requires 12 address lines.

Q2. System bus like have - Data bus
- Address bus
- Control bus

* Bidirectional Bus is Data bus (allowing data transfer both to and from CPU)

Q3. Microcomputer: is small, scale computer system built around single chip central processing unit (microprocessor), containing memory (RAM/ROM) and input/output (I/O) interface.

Example: - personal Computer (PC)

- Raspberry Pi

- Desktop

Q4. It requires 16 lines if referring specifically to the data bus in 16-bit microprocessor

Q5. component of an Assembly language program.

1. Label field

2. Opcode (Mnemonic) field.

3. operand field

4. Comment field

* optional component is label field and comment field.

Q6. the command provided is `lcd.print("52, BIN");`

Because "52, BIN" is enclosed in double quotes as a string literal, the LCD will display the exact text string.

It would output the binary representation of 52, which is 110100

$$52_{10} = (110100)_2.$$

Q7. Arduino board (like the Uno) use an 8-bit pulse width modulation timer for simulated analog output using the `analogWrite()` function.

Since $2^8 = 256$ possible discrete values, the range of duty cycle spans from 0 to 255.

Range: 0 - 255

Q8. `if (condition) {`

 action 1

`}`

`else if (condition2) {`

 action 2

`}`

`else {`

 action 3

`}`

Q9. - Data Transfer Group

Example: `MOV A, B`

- Arithmetic Group

Example: `ADD B`

3. Logical Group.

Example: ANA B

4. Branch Groups

Example: JMP 2000H

5. Stack, I/O and Machine Control Group.

Example: HLT.

Q10. instruction: XRA E performs a bitwise exclusive-OR (XOR) between the Accumulator (A) and register, E, storing the result A.

Initial Contents

$$A = EEH = 1110\ 1110_2$$

$$E = 5AH = 0101\ 1010_2$$

XOR operation:

$$\begin{array}{r} A: 1110\ 1110 \\ E: 0101\ 1010 \\ \hline 1011\ 0100 \Rightarrow B4H \end{array}$$

Final Register Contents

$$A = B4H$$

E = 5AH (Register E remains unchanged)

Q11. Analysis of code

- When pin2 reads 1 (5V), pin1 is set Low (0V)
- When pin2 reads 0 (0V), pin1 is set High (5V)
- Thus, pin1 acts as an inverted Signal (Not gate) of pin2
- Sequence sent of pin2: 1011
- Output Generated on pin1: 0100

Q12. TWO TYPES of 7-segment Displays.

- common Anode (CA): all positive terminals (anodes) of the LEDs are connected together to Vcc (or). Individual segment turn ON when supplied with a low (0) logic signal.

• Common cathode (CC): all negative terminals (cathodes) of the LEDs are connected together to ground (GND). Individual segment turns on when supplied with HIGH (1) logic signal.

Number Displayed segment active (HIGH): A, C, D, F, G

Standard 7-segment Segment arrangement.

Top: A

Top Right: B

Bottom-Right: C

Bottom: D

Bottom-left: E

Top-left: F

Middle: G

Lighting segment A, C, D, F, G creates the digital character "5".

Q13. A. initialize serial communication

⇒ serial.begin(9600);

B. set cursor in 2nd row and 4th column of an LCD;

⇒ lcd.setCursor(3, 1);

C. Read Humidity float value from DHT11;

float dht = TEMP.readHumidity();

D. Read pressed key from keypad

char key = KEYT.getKey();

Q14. A. Full word: Radio Frequency Identification.

B. three components of an RFID system:

1. RFID Reader/Writer (Transceiver/Module)

2. RFID Tag/Card (Transponder)

3. Processing Device/Host Controller (e.g. microcontroller/Arduino)

4. Antenna

(C). Passive RFID Tags; Have no internal power source/battery. They are powered by the electromagnetic energy transmitted from the RFID reader.

- Active RFID tags: contains their own onboard power sources (battery) to continuously transmits signal over longer distances.

Do Technical names For Component in the Image.

- * (1): RC522 RFID Reader Module (or Antenna / transceiver module)
- * (2): Key Fob RFID Tag
- * (3): Smart card / RFID Proximity Card tag

(15)

1. From the system called Automated warehouse Gate Control System

Requirements components

- Arduino Uno
- Ultrasonic sensor
- Motor Driver L298N.
- Demotor
- LCD 16x2 display
- LED indicator
- Limit switches (x2)
- Power supply

Arduino codes

```
#include <Wire.h>
```

```
#include <LiquidCrystal - I2C.h>
```

```
LiquidCrystal - I2C lcd (0x27, 16, 2);
```

```
#define TRIG 9
```

```
#define ECHO 10
```

```
#define ENA 5
```

```
#define IN1 6
```

```
#define IN2 7.
```

```
#define LED - GREEN A0
```

```
#define LED - RED A1
```

```
#define LIMIT - OPEN 2
```

```
#define LIMIT - CLOSE 3
```

```
long duration;  
int distance;  
int motorspeed;
```

```
void Setup() {
```

```
pinMode(TRIG, OUTPUT);  
pinMode(ECHO, INPUT);  
pinMode(ENA, OUTPUT);  
pinMode(IN1, OUTPUT);  
pinMode(IN2, OUTPUT);  
pinMode(LED_GREEN, OUTPUT);  
pinMode(LIMIT_OPEN, INPUT_PULLUP);  
pinMode(LIMIT_CLOSE, INPUT_PULLUP);
```

```
Lcd.init();  
Lcd.backlight();  
Lcd.setCursor(0, 0);  
Lcd.print("Warehouse Gate");  
delay(2000);  
}
```

```
void loop() {
```

```
digitalWrite(TRIG, LOW);  
delayMicroseconds(2);  
digitalWrite(TRIG, HIGH);  
delayMicroseconds(10);  
digitalWrite digitalWrite(TRIG, LOW);
```

```
duration = pulseIn(ECHO, HIGH);  
distance = duration * 0.034 / 2  
// map distance (20 - 150cm) to  
motor speed (255 - 100) → close = faster  
motor speed = map(distance, 20, 150, 255, 100);
```

```
Lcd.clear();  
Lcd.setCursor(0, 0);  
Lcd.print("Distance:");  
Lcd.print(distance);  
Lcd.print(" cm");
```

```
// check limit switch first
```

```
if (digitalRead(LIMIT_OPEN) == LOW) {
```



```

analogWrite(ENA, 0);
digitalWrite(LED - RED, LOW);
lcd.setCursor(0, 1);
lcd.print("Gate fully closed");
return;
}

```

```

if (distance > 50) {

```

// Gate opening

```

digitalWrite(IN2, HIGH);
digitalWrite(IN2, LOW);
digitalWrite(ENA, motorSpeed);
digitalWrite(LED - GREEN, HIGH);
digitalWrite(LED - RED, LOW);

```

```

lcd.setCursor(0, 1);
lcd.print("Gate opening ...");
}

```

```

else {

```

// Gate closing

```

digitalWrite(IN1, LOW);
digitalWrite(IN2, HIGH);
analogWrite(ENA, 180); // fixed
closing speed.

```

```

digitalWrite(LED - GREEN, LOW);
digitalWrite(LED - RED, HIGH);

```

```

lcd.setCursor(0, 1);
lcd.print("Gate closing ...");
}

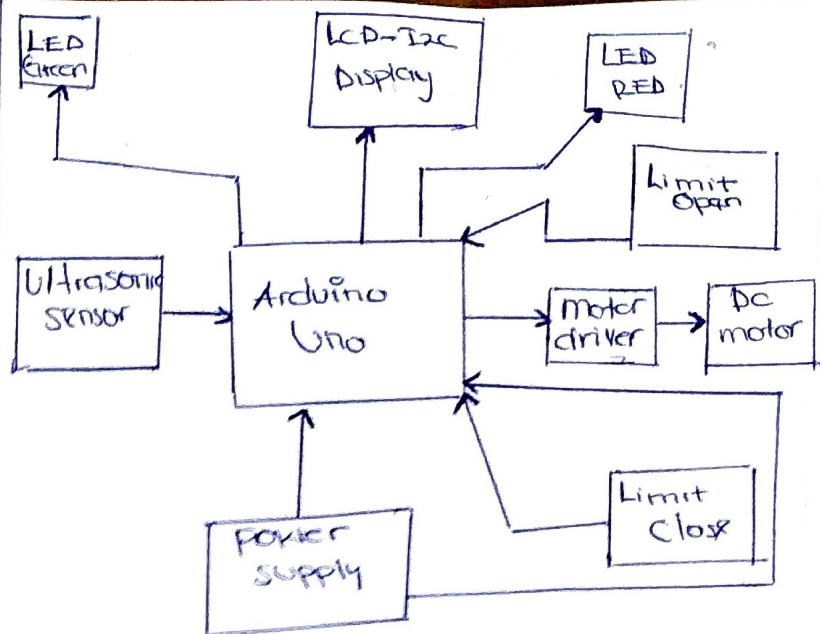
```

```

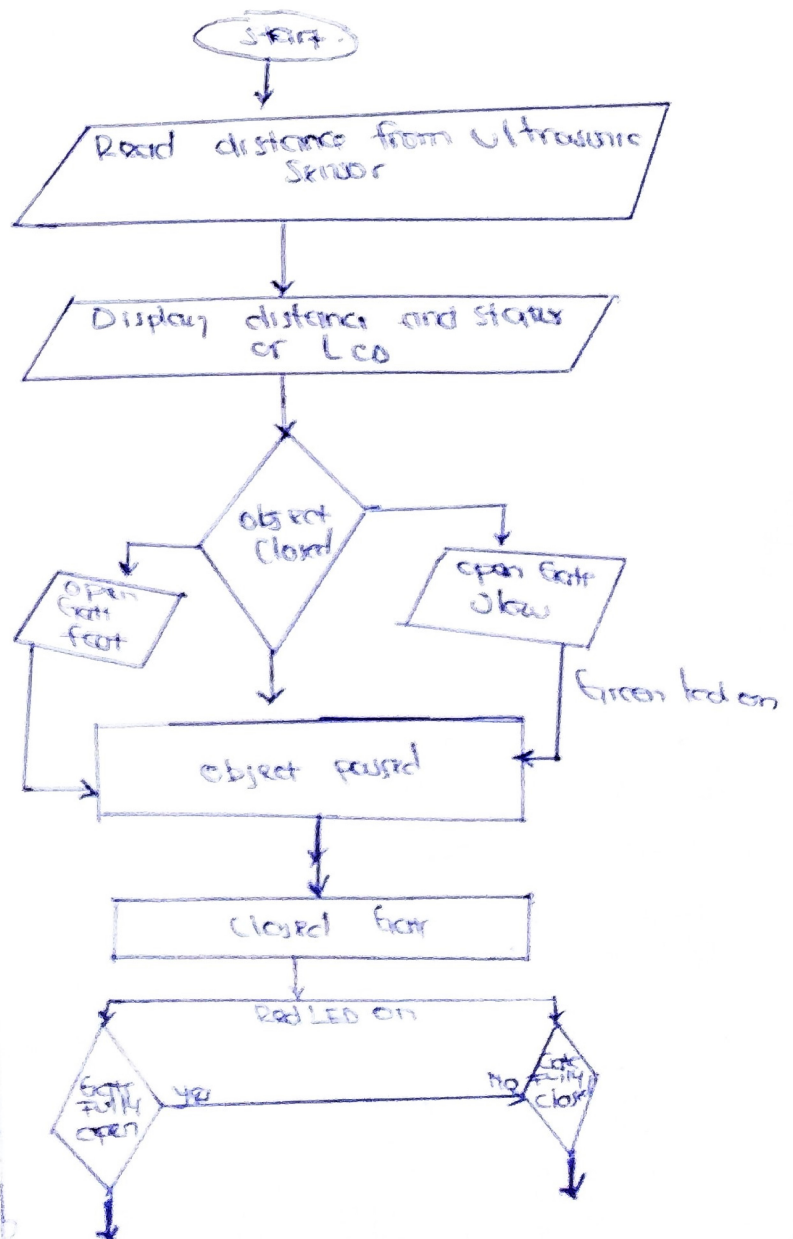
}

```

Block diagram for the system.



Flowchart for the system.



Stop motor
"Gettting fully open"

Stop motor
"Gettting fully"
closed

Continue
operation

② From the system smart Automatic indoor and outdoor lighting control system.

Component Required

- Arduino uno
- LDR sensor
- PIR sensor
- Relay module
- outdoor lamp
- indoor lamp
- power supply

Arduino code

```
#define LDR A0
#define PIR 8
#define RELAY-OUTDOOR 2
#define RELAY-INDOOR 3

int ldrValue;
int pirValue;
```

```
void setup() {
```

```
  pinMode(RELAY-OUTDOOR, OUTPUT);
  pinMode(RELAY-INDOOR, OUTPUT);
  pinMode(PIR, INPUT);
  Serial.begin(9600);
}
```

```
void loop() {
```

```
  // Read sensors
  ldrValue = analogRead(LDR);
  pirValue = digitalRead(PIR);
```

```
// outdoor lighting control
```

```
if (ldrValue < 200) {
```

```
  // darkness threshold
```

```
  digitalWrite(RELAY-OUTDOOR, HIGH);
}
```

```
else {
```

```
  digitalWrite(RELAY-OUTDOOR, LOW);
}
```

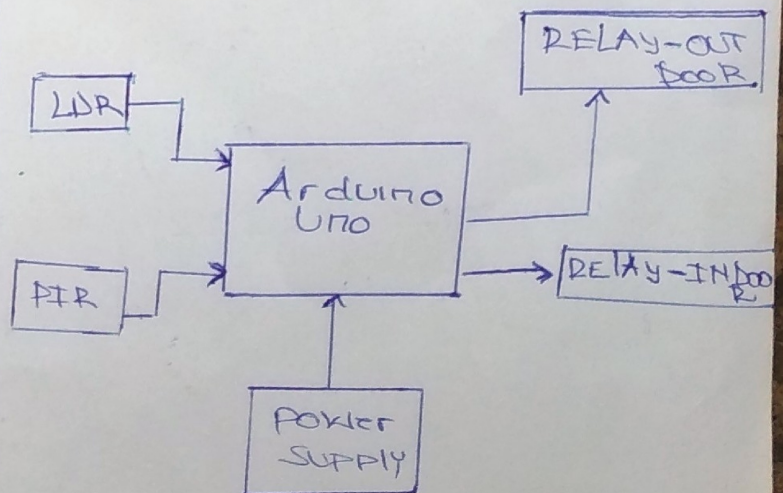
```
if (pirValue == HIGH) {
```

```
  digitalWrite(RELAY-INDOOR, HIGH);
```

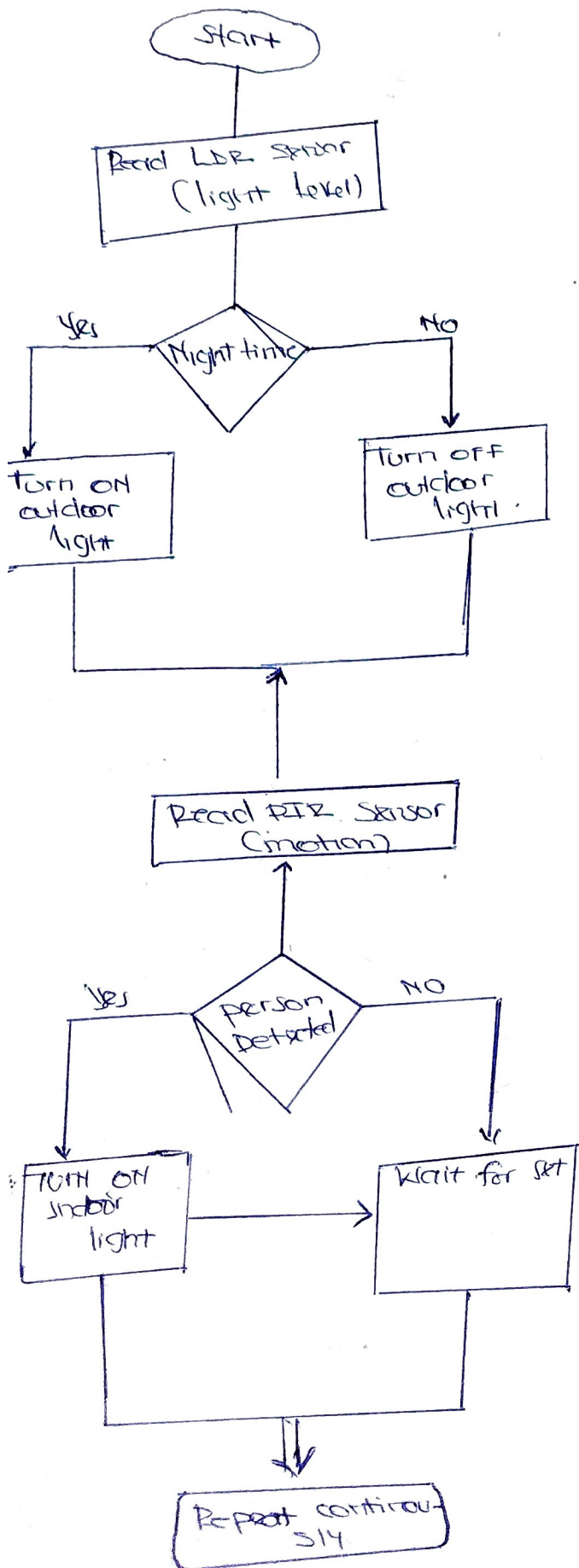
```
  // turn on indoor lamp
```

```
  delay(500);
}
```

Block diagram for the system



Control Flowchart for the System.



Required Components

- ⇒ Arduino uno
- Ultrasonic Sensors HC-SR04
- LCD 16x2 I2C Display
- Buzzer
- LED indicator
- Relay module
- power supply.

Arduino code for the System.

```

#include <Wire.h>
#include <liquidcrystal-I2C.h>
liquidcrystal-I2C lcd(0x27, 16, 2);

#define TRIG - ENTRANCE 9
#define ECHO - ENTRANCE 8
#define TRIG - EXIT 7
#define ECHO - EXIT 6

#define BUZZER 4
#define LED - GREEN 5
#define LED - RED 3

int count = 0;
const int maxCapacity = 10;
long duration;
int distance Entrance, distance Exit;

void Setup() {
  pinMode(TRIG - ENTRANCE, OUTPUT);
  pinMode(ECHO - ENTRANCE, INPUT);
  pinMode(TRIG - EXIT, INPUT);
  pinMode(ECHO - EXIT, INPUT);
  pinMode(LED - GREEN, OUTPUT);
  pinMode(LED - RED, OUTPUT);
}

lcd.begin();
lcd.backlight();
lcd.setCursor(0, 0);
lcd.print("Parking system");
delay(2000);
  
```

```
int getDistance (int trig pin, int echopin)
```

```
{
    digitalWrite (trigpin, LOW);
    delay Microseconds (20);
    digitalWrite (trigpin, HIGH);
    delay Microseconds (200);
    digitalWrite (trigpin, LOW);
```

```
duration = pulseIn (echopin, HIGH);
    return duration * 0.034 / 2;
```

```
// distance in cm.
```

```
}
```

```
void loop () {
```

```
    distance Entrance = getDistance (TRIG-Entrance, ECHO-ENTRANCE);
```

```
    distance Exit = getDistance (TRIG-Exit, ECHO-EXIT);
```

```
    lcd.clear ();
```

```
    lcd.setCursor (0, 0);
```

```
    lcd.print ("cars:");
```

```
    lcd.print (count);
```

```
    lcd.setCursor (0, 1);
```

```
    lcd.print ("free: ");
```

```
    lcd.print (max-capacity-count);
```

```
// Vehicle entering (threshold < 15cm)
```

```
if (distance Entrance < 15 || count < max-capacity) { count++;
```

```
    delay (1000);
```

```
} // Vehicle existing
```

```
if (distance Exit < 15 || count > 0)
```

```
{ count--;
```

```
    delay (1000);
```

```
}
```

```
// Full capacity warning
if (count >= MAX-CAPACITY) {
```

```
    digitalWrite (BUZZER, HIGH);
    digitalWrite (LED-RED, LOW);
```

```
    lcd.setCursor (0, 1);
```

```
    lcd.print ("parking full:");
```

```
}
```

```
else
```

```
{
```

```
    digitalWrite (BUZZER, LOW);
```

```
    digitalWrite (LED-RED, LOW);
```

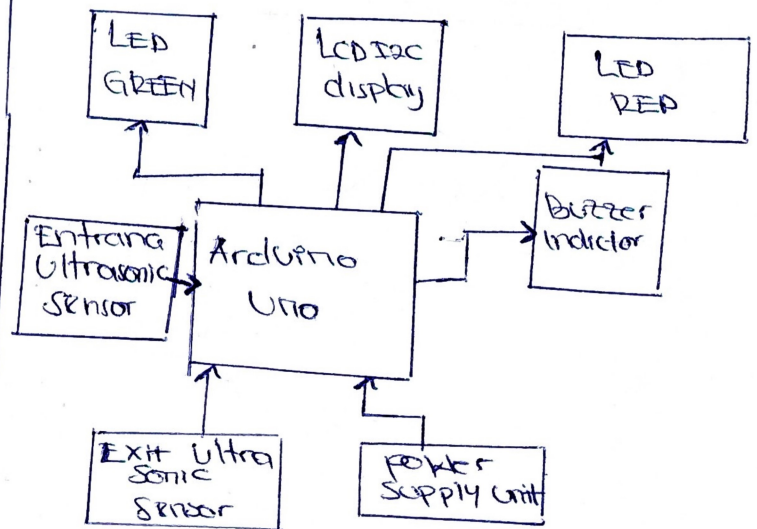
```
    digitalWrite (LED-GREEN, HIGH);
```

```
}
```

```
    delay (500);
```

```
}
```

Block diagram For the system



Flow chart for car system

