Write the Pseudocode and Flowchart for the problem statements mentioned below:

-

1. Smart Home Temperature Control

Problem Statement:

Design a temperature control system for a smart home. The system should read the current temperature from a sensor every minute and compare it to a user-defined setpoint.

Requirements:

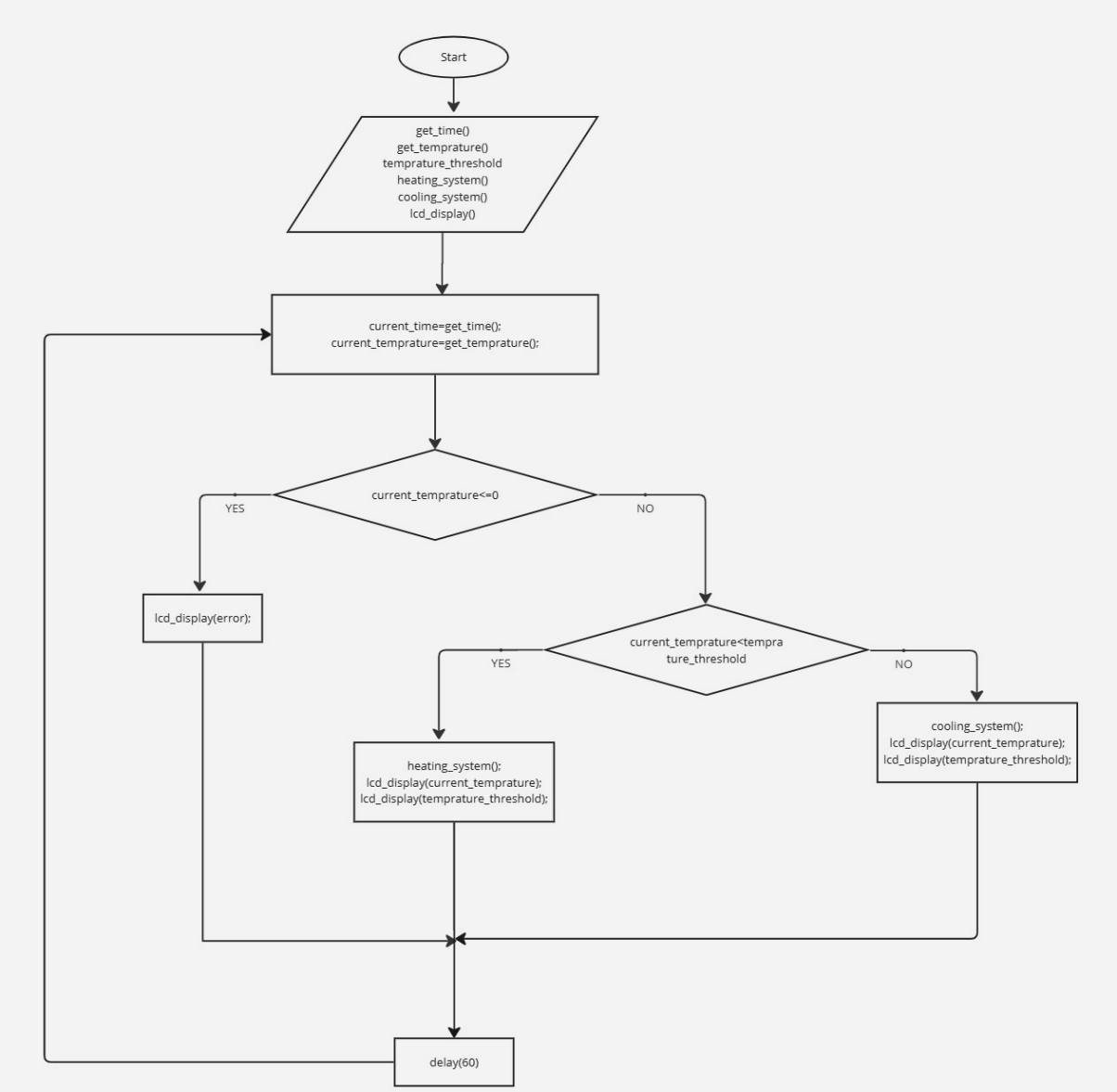
- If the current temperature is above the setpoint, activate the cooling system.
- If the current temperature is below the setpoint, activate the heating system.
- Display the current temperature and setpoint on an LCD screen.
- Include error handling for sensor failures.

Pseudocode

```
//get inputs
get_time();
get_temprature();
temperature_threshold;
delay();
//set outputs
heating_system();
cooling_system();
lcd_display();
main{
while(1){
current_time=get_time();
current_temprature=get_temprature();
if(current_temprature<=0;){
lcd_display(error);
 }
else{
if(current_temprature<temprature_threshold){
                    heating_system();
                    lcd_display(current_temprature);
                    lcd_display(temprature_threshold);
}
else if(current_temprature>temprature_threshold){
          cooling_system();
                    lcd_display(current_temprature);
                    lcd_display(temprature_threshold);
}
   }
delay(60);
  }
}
delay(){
return seconds;
}
get_time(){
 return time;
}
get_temprature(){
```

return temprature;
}
heating_system(){
activate heating system;
}
cooling_system(){
activate cooling system;
}

Flowchart



****

2. Automated Plant Watering System

Problem Statement:

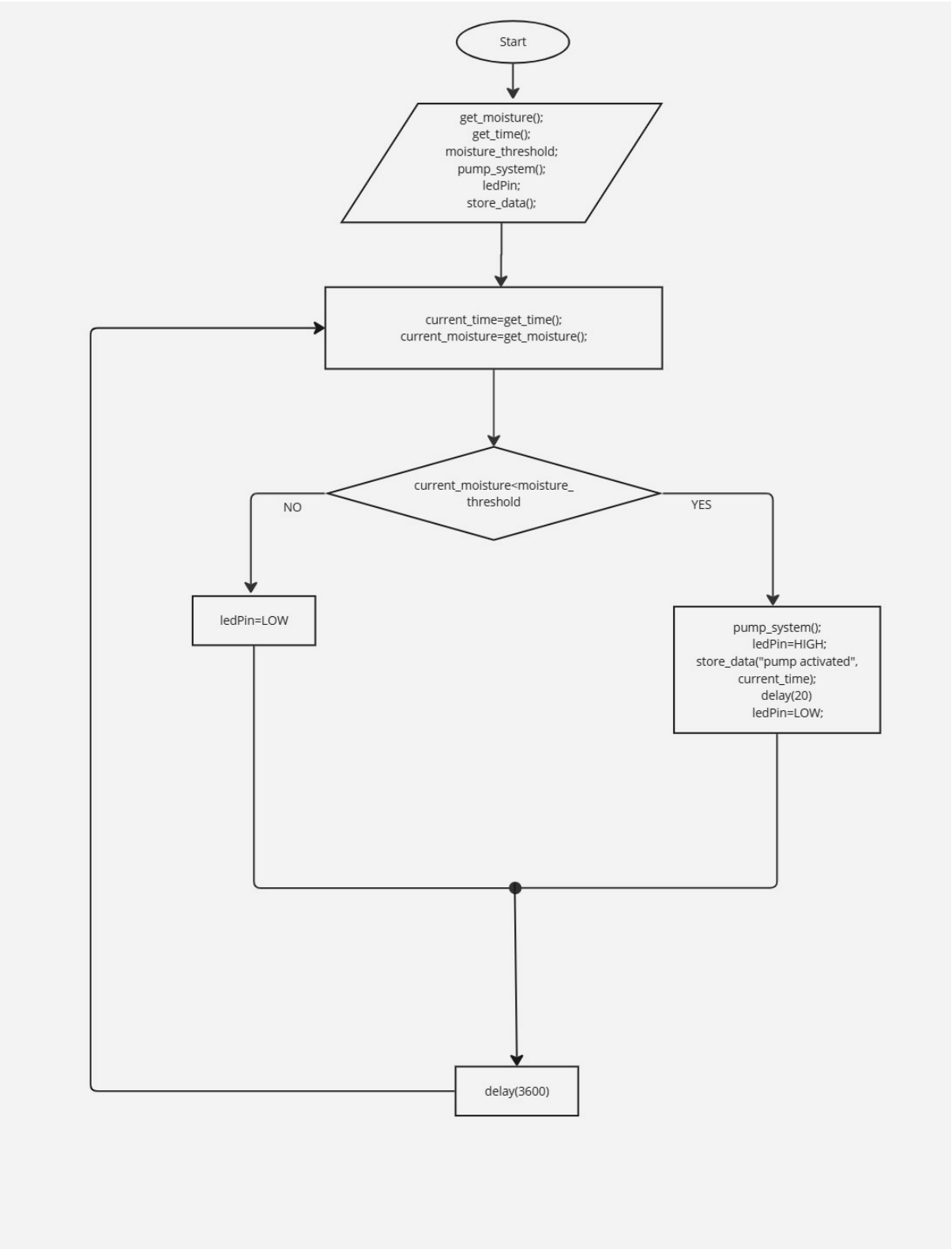
Create an automated watering system for plants that checks soil moisture levels and waters the plants accordingly.

Requirements:

- Read soil moisture level from a sensor every hour.
- If moisture level is below a defined threshold, activate the water pump for a specified duration.
- Log the watering events with timestamps to an SD card.
- Provide feedback through an LED indicator (e.g., LED ON when watering).

```
//get inputs
get_moisture();
get_time();
moisture_threshold;
delay();
//set outputs
pump_system();
ledPin;
store_data();
main{
while(1){
current_time=get_time();
current_moisture=get_moisture();
if(current_moisture<moisture_threshold){
            pump_system();
      ledPin=HIGH;
      delay(20);
store_data("pump activated", current_time);
            ledPin=LOW;
  }
else{
            ledPin=LOW;
  }
    }
delay(3600);
  }
}
delay(){
return seconds;
}
get_time(){
 return time;
}
get_moisture(){
 return moisture;
}
pump_system(){
activate pump system;
}
store_data(pump_status,timestamp){
pump_status;
timestamp;
}
```

Flowchart



3. Motion Detection Alarm System

Problem Statement:

Develop a security alarm system that detects motion using a PIR sensor.

Requirements:

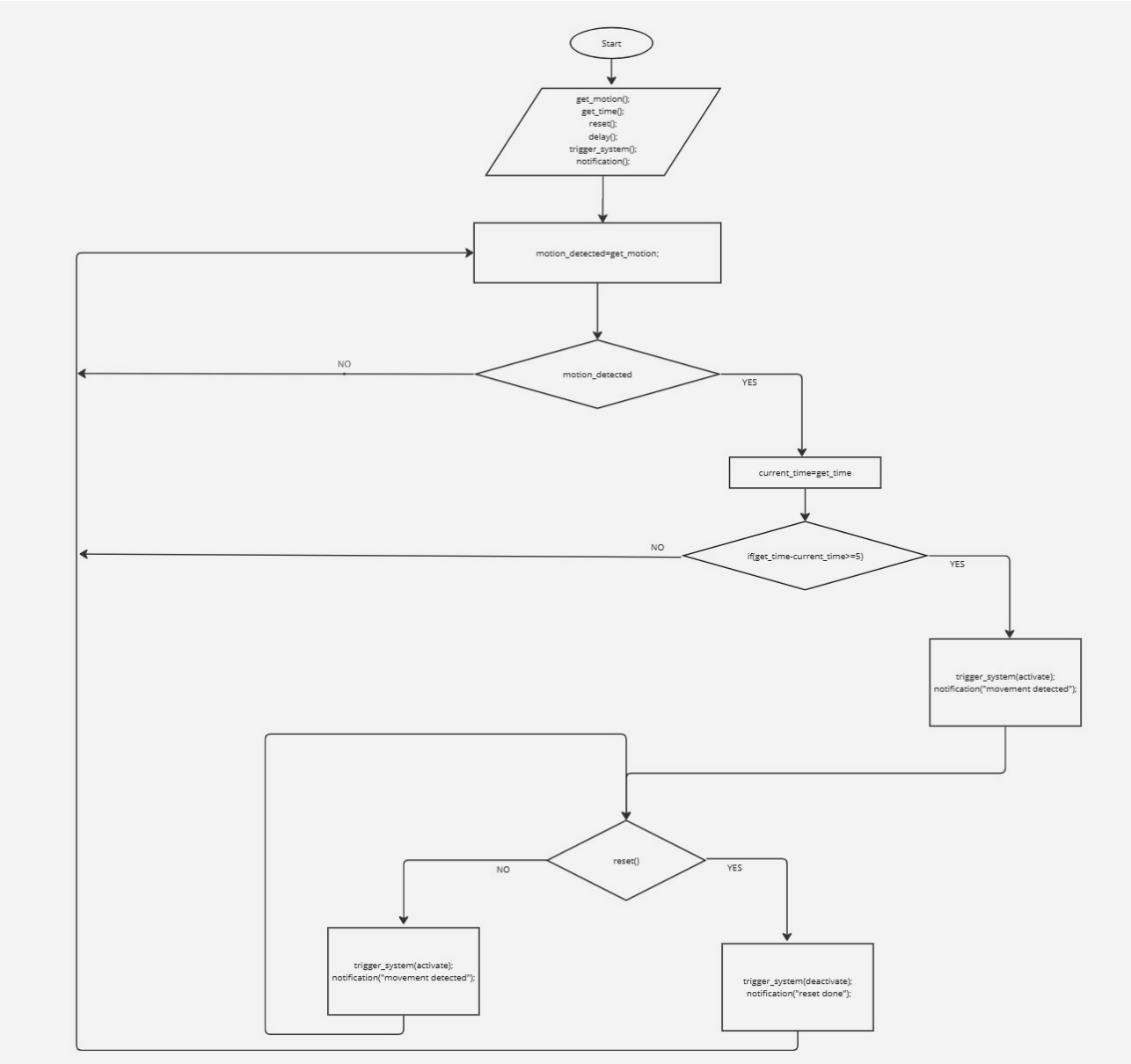
- Continuously monitor motion detection status.
- If motion is detected for more than 5 seconds, trigger an alarm (buzzer).
- Send a notification to a mobile device via UART communication.
- Include a reset mechanism to deactivate the alarm.

Pseudocode

```
 //get inputs
get_motion();
get_time();
reset();
delay();
//set outputs
trigger_system();
notification();
();
main{
while(1){
motion_detected=get_motion();

if(motion_detected){
      current_time=get_time();
           while(motion_detected){
          if(get_time-current_time>=5){
             trigger_system(activate);
             notification("movement detected");
             break;
                }
       }
  }
if(reset()){
 trigger_system(deactivate);
 notification("reset done");
}
}
delay(){
return seconds;
}
get_time(){
 return time;
}
get_motion(){
 return motion;
}
trigger_system(){
activate trigger system;
}
notification(){
return motification;
}
reset(){
stop trigger;
stop notification;
}
```

Flowchart



4. Heart Rate Monitor

Problem Statement:

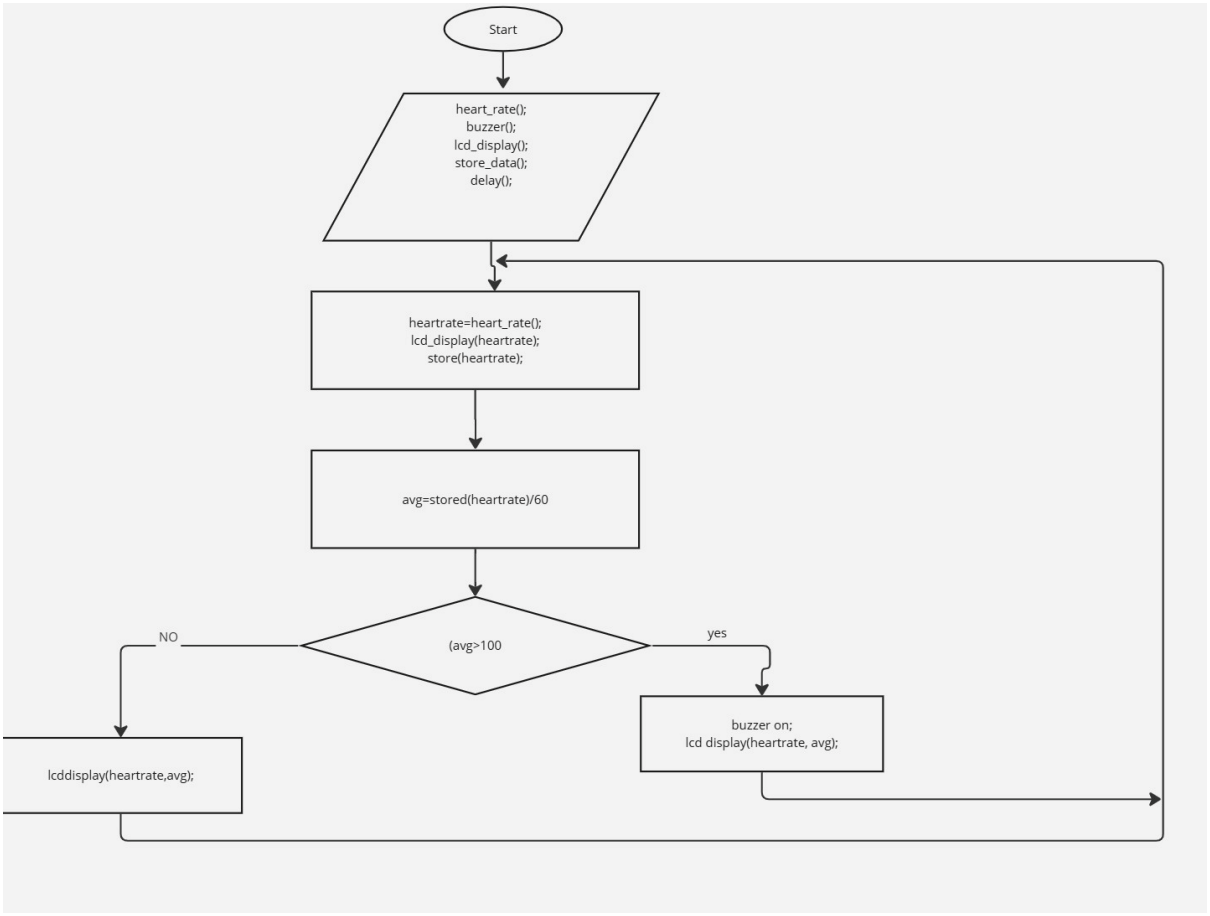
Implement a heart rate monitoring application that reads data from a heart rate sensor.

Requirements:

- Sample heart rate data every second and calculate the average heart rate over one minute.
- If the heart rate exceeds 100 beats per minute, trigger an alert (buzzer).
- Display current heart rate and average heart rate on an LCD screen.
- Log heart rate data to an SD card for later analysis.

```
heart_rate();
buzzer();
lcd_display();
store_data();
delay();
while(1){
heartrate=heart_rate();
lcd_display(heartrate);
store(heartrate);
```

```
delay(60);
avg=stored(heartrate)/60
if(avg>100){
buzzer on;
lcd display(heartrate, avg);
}
else{
lcddisplay(heartrate,avg);
}
```



5. LED Control Based on Light Sensor

Problem Statement:

Create an embedded application that controls an LED based on ambient light levels detected by a light sensor.

Requirements:

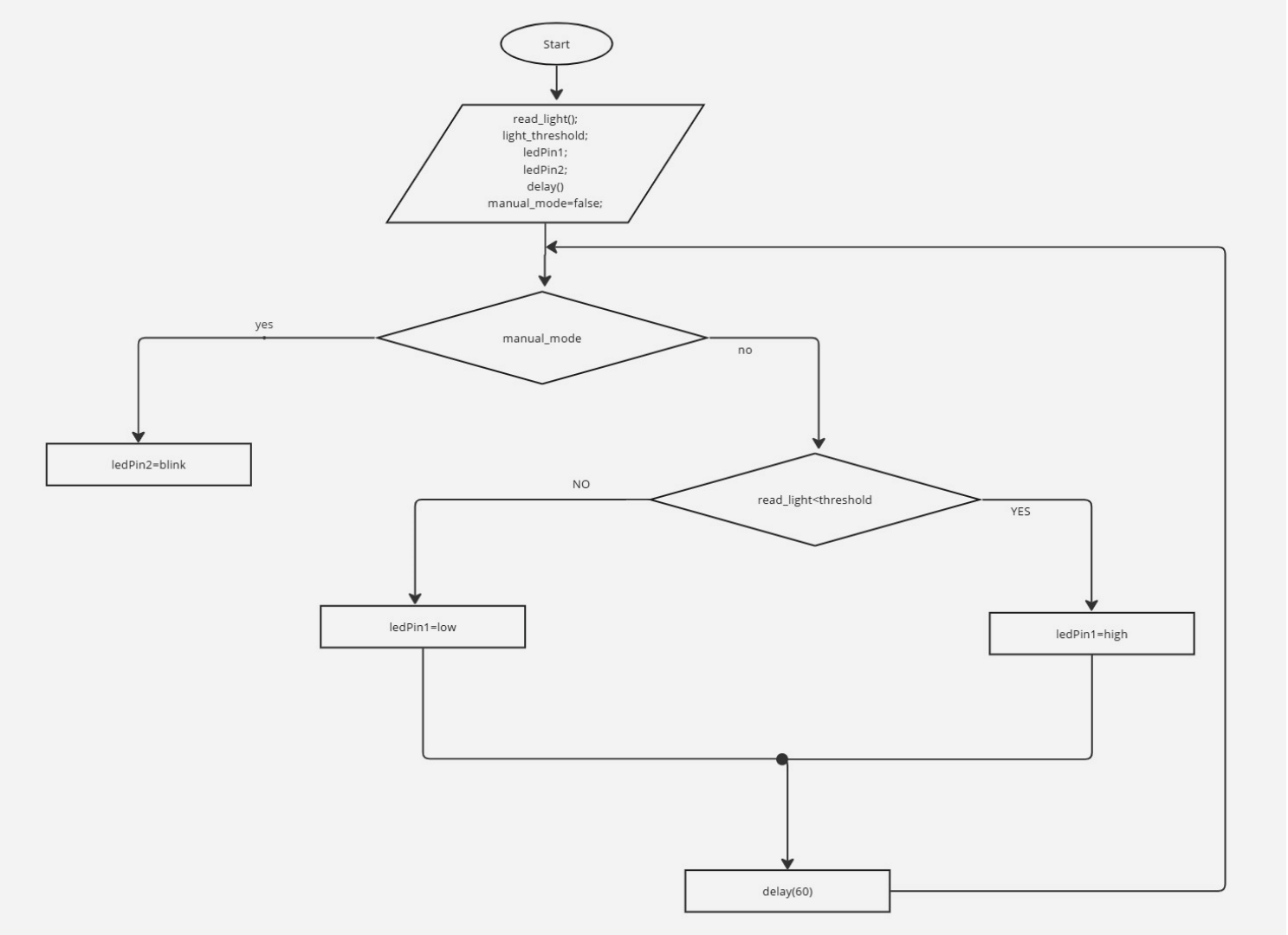
- Read light intensity from the sensor every minute.
- If light intensity is below a certain threshold, turn ON the LED; otherwise, turn it OFF.
- Include a manual override switch that allows users to control the LED regardless of sensor input.

- Provide status feedback through another LED (e.g., blinking when in manual mode).

```
read_light();
light_threshold;
ledPin1;
ledPin2;
delay()
manual_mode=false;
main(){
while(1){
if(manual_mode==true){
  ledPin2=blink;

}
else if(manual_mode==false){
if(read_light<threshold){
ledPin1=high;
}
else{
ledPin1=low;
}
delay(60);
}

}
read_light(){
return light_value;
}
delay(){
return seconds;
}
```



6. Digital Stopwatch

Problem Statement:

Design a digital stopwatch application that can start, stop, and reset using button inputs.

Requirements:

- Use buttons for Start, Stop, and Reset functionalities.
- Display elapsed time on an LCD screen in hours, minutes, and seconds format.
- Include functionality to pause and resume timing without resetting.
- Log start and stop times to an SD card when stopped.

```
button_read();
start()
stop()
reset()
get_time;
lcd_display()
counter;
count=0;
data();
while(1){
process=button_read();
```

switch process

1: "start"

1.a start();

1.b counter start;

1.c count++;

1.d if(count==2){

counter pause;

}

1.e else if(count==3){

counter restart;

count=0;

}

1.f store(counter);

1.g lcd_display(hr,min,sec);

2: "stop"

2.a stop();

2.b counter stoped;

2.c lcd_display(hr,min,sec)

2.d store(counter)

3: "reset"

3.a reset();

3.b counter reset;

3.3 lcd_display(hr,min,sec)

7. Temperature Logging System

Problem Statement:

Implement a temperature logging system that records temperature data at regular intervals.

Requirements:

- Read temperature from a sensor every 10 minutes.
- Store each reading along with its timestamp in an array or log file.
- Provide functionality to retrieve and display historical data upon request.
- Include error handling for sensor read failures.

8. Bluetooth Controlled Robot

Problem Statement:

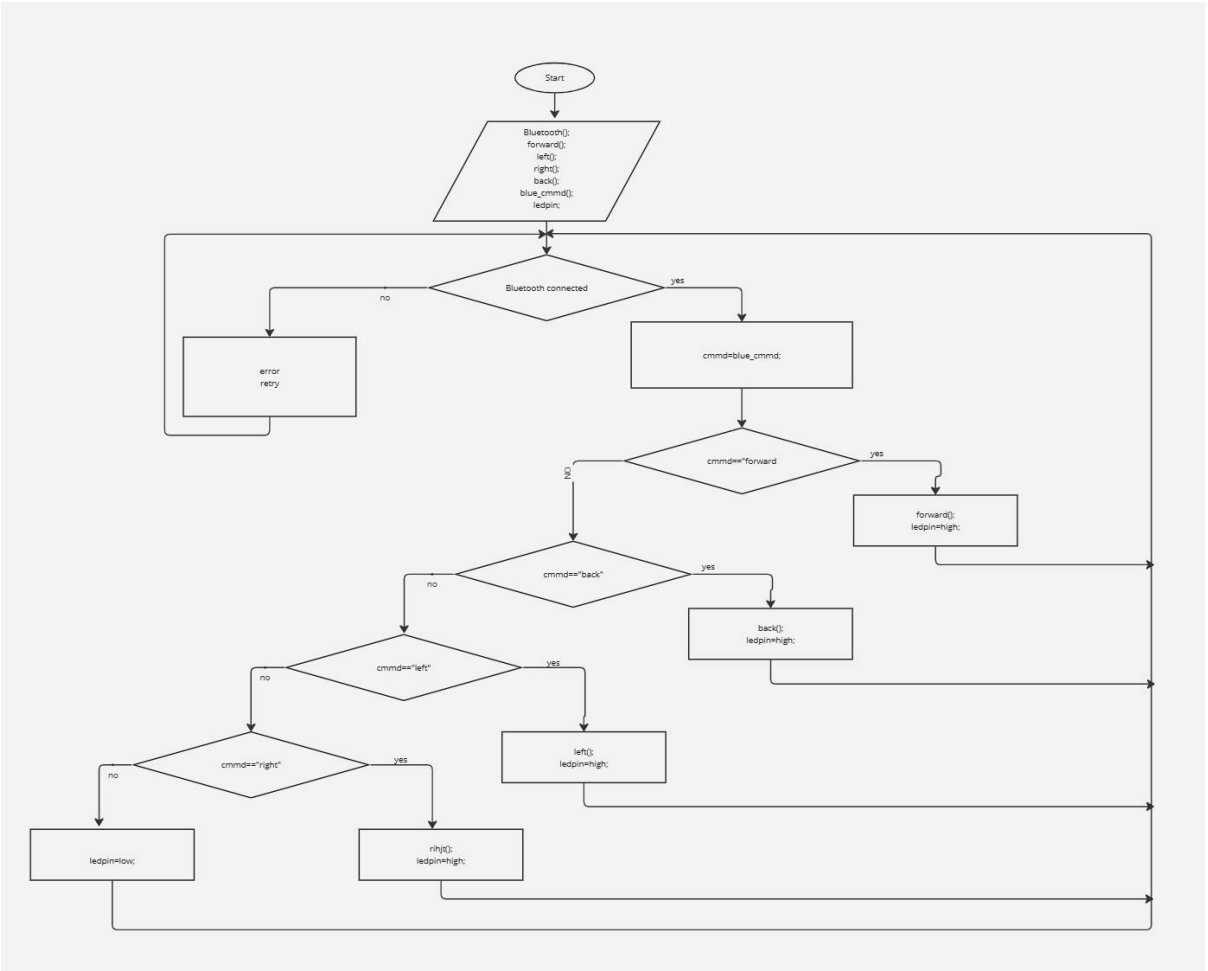
Create an embedded application for controlling a robot via Bluetooth commands.

Requirements:

- Establish Bluetooth communication with a mobile device.
- Implement commands for moving forward, backward, left, and right.
- Include speed control functionality based on received commands.

- Provide feedback through LEDs indicating the current state (e.g., moving or stopped).

```
Bluetooth();
forward();
left();
right();
back();
blue_cmmd();
ledpin;
 while(1){
if Bluetooth connected{
 cmmd=blue_cmmd;
 if cmmd=="forward{
 forward();
ledpin=high;
}
else if  cmmd=="back"{
back();
ledpin=high;
}
else if  cmmd=="left"{
left();
ledpin=high;
}
else if  cmmd=="right"{
right();
ledpin=high;
}
else{
ledpin=low;
}
}
}
```



9. Battery Monitoring System

Problem Statement:

Develop a battery monitoring system that checks battery voltage levels periodically and alerts if voltage drops below a safe threshold.

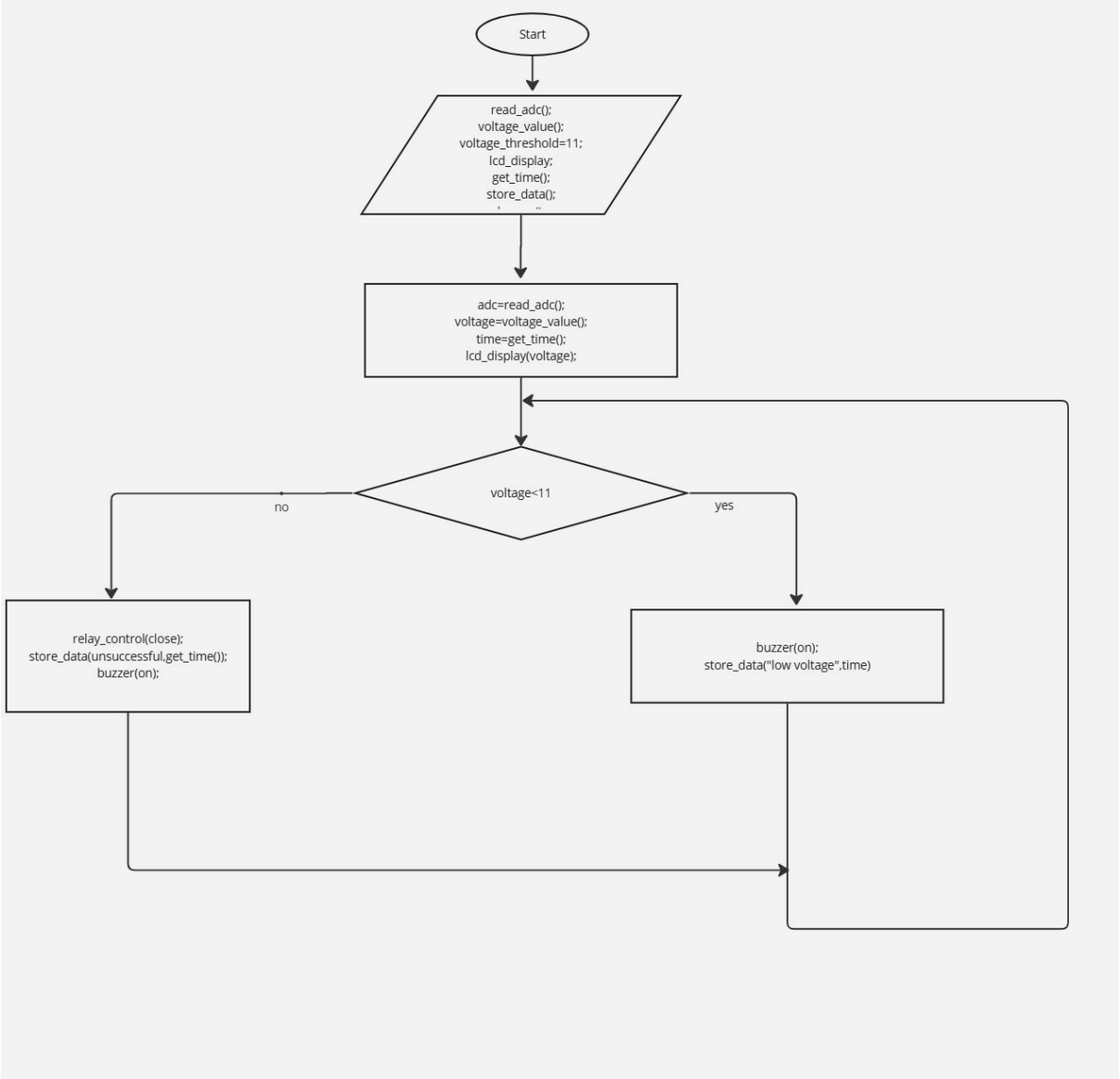
Requirements:

- Measure battery voltage every minute using an ADC (Analog-to-Digital Converter).
- If voltage falls below 11V, trigger an alert (buzzer) and log the event to memory.
- Display current voltage on an LCD screen continuously.
- Implement power-saving features to reduce energy consumption during idle periods.

```
read_adc();
voltage_value();
voltage_threshold=11;
lcd_display;
get_time();
store_data();
buzzer();
while(1){
adc=read_adc();
voltage=voltage_value();
time=get_time();
lcd_display(voltage);
if(voltage<11){
buzzer(on);
store_data("low voltage",time);
```

}

}



10. RFID-Based Access Control System

Problem Statement:

Design an access control system using RFID technology to grant or deny access based on scanned RFID tags.

Requirements:

- Continuously monitor for RFID tag scans using an RFID reader.
- Compare scanned tags against an authorized list stored in memory.
- Grant access by activating a relay if the tag is authorized; otherwise, deny access with an alert (buzzer).
- Log access attempts (successful and unsuccessful) with timestamps to an SD card.

rfid_scanner();

data_list;

relay_control;

store_data();

buzzer();

get_time();

main(){

```
while(1)
{
if(rfid_scan match with data_list){
relay_control(open);
store_data(successful,get_time());



}
else{
relay_control(clos);
store_data(unsuccessful,get_time());
buzzer(on);
}
}
}
```