MaRS CLUB

Task #01

Om Kinage

EC22B1111

**Project 1: Ambient Monitoring System**

* **Tinkercad link:** <https://www.tinkercad.com/things/jzUkemimsYF>
* **Code:**

#include <LiquidCrystal.h>

// Initialize the LCD display:

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// Initialize the sensor pins:

int lightSensor = A0;

int moistureSensor = A1;

int temperatureSensor = A2;

void setup()

{

// Set the ambient light sensor as input

pinMode(lightSensor, INPUT);

pinMode(moistureSensor, INPUT);

pinMode(temperatureSensor, INPUT);

// Initialize the LCD display (16x2):

lcd.begin(16, 2);

}

void loop()

{

// Read the ambient light sensor value:

int lightValue = analogRead(lightSensor);

// Check if the light is bright enough:

if (lightValue > 200)

{

// Read the soil moisture & temperature sensor value

int moistureValue = analogRead(moistureSensor);

int temperatureValue = analogRead(temperatureSensor);

// Set cursor to first column and first row of LCD display:

lcd.setCursor(0, 0);

//Print Moisture value on first line of LCD display:

lcd.print("Moisture: ");

lcd.print(moistureValue);

// Set cursor to first column and second row of LCD display:

lcd.setCursor(0, 1);

//Print Temperature value on first line of LCD display:

lcd.print("Temperature: ");

lcd.print(temperatureValue);

}

else if(lightValue <= 200)

{

// Set cursor to first column and first row:

lcd.setCursor(0, 0);

lcd.print("not\_enough\_light");

delay(1000);

//Clear the LCD screen:(after some time delay)

lcd.clear();

}

}

* **Explanation:**

The given code represents an Arduino project that serves as an ambient monitoring system. It utilizes various sensors to measure and display information about the environment, specifically the soil moisture and temperature. The system also includes an ambient light sensor to determine if the light conditions are sufficient for the measurements to be accurate.

* **Purpose:**

The purpose of this project is to monitor and display relevant environmental data, such as soil moisture and temperature, using an LCD display. The system continuously reads the ambient light sensor and checks if the light level is above a certain threshold (200 in this case). If the light is bright enough, it proceeds to read the soil moisture and temperature values from their respective sensors. The measurements are then displayed on a 16x2 LCD display.

The system is useful in applications where monitoring and maintaining optimal soil conditions are necessary, such as agriculture, gardening, or plant research. By having real-time information about the soil moisture and temperature, users can make informed decisions regarding irrigation, plant health, or climate control. Additionally, the system provides feedback in case the light conditions are inadequate, which can help users identify if their plants are not receiving enough light.

**Project 2: PIR Sensor Motion Detection**

* **Tinkercad link:** <https://www.tinkercad.com/things/9PnY97JaRa6>
* **Code:**

#include "LiquidCrystal.h" //Library of lcd

LiquidCrystal lcd(10,9,8,7,6,5); //pin of lcd

int val;

void setup()

{

Serial.begin(9600);

lcd.begin(16,2);

lcd.setCursor(0,0);

lcd.print(" PIR Sensor ");

pinMode(13,INPUT); // pir sensor output pin connected

}

void loop()

{

val = digitalRead(13); // pir sensor output pin connected

Serial.println(val); // see the value in serial monitor in Arduino IDE

//delay(100);

if(val == 1 )

{

lcd.setCursor(0,1);

lcd.print(" DETECTED ");

delay(2000);

}

else

{

lcd.setCursor(0,1);

lcd.print(" NOT DETECTED ");

}

}

* **Explanation:**

The provided code represents an Arduino project that utilizes a Passive Infrared (PIR) sensor to detect motion. It displays the detection status on an LCD screen.

* **Purpose:**

The purpose of this project is to detect motion using a PIR sensor and provide visual feedback on an LCD display. The PIR sensor is connected to pin 13 of the Arduino board and is responsible for detecting changes in infrared radiation, typically caused by the movement of objects or living beings within its range.

Upon detecting motion, the PIR sensor outputs a high signal (1) to the Arduino board. The code continuously reads the value from the sensor using the digitalRead () function. If the value is 1, indicating motion is detected, the LCD display shows the message "DETECTED" on the second line. This visual indication helps users know when the PIR sensor detects motion.

If no motion is detected, the PIR sensor outputs a low signal (0), and the LCD display shows the message "NOT DETECTED" on the second line.

The code also includes a serial communication line (Serial.println (Val)) to display the sensor output value in the Arduino IDE's serial monitor. This can be useful for debugging and monitoring the PIR sensor's behavior.

This project can be applied in various applications where motion detection is required, such as security systems, automated lighting systems, or interactive installations. By using the PIR sensor and the LCD display, users can easily observe and respond to the presence or absence of motion in their environment.

**Project 2: Sensor Monitoring with Tilt Detection**

* **Tinkercad link:** <https://www.tinkercad.com/things/jYvDIY8iGqZ>
* **Code:**

//Force and flex sensor pin(analog i/p):

int forceSensorPin = A0;

int flexSensorPin = A1;

//Tilt sensor pin and buzzer(digital i/p):

int tiltSensorPin = 2;

int buzzerPin = 3;

void setup()

{

// Set tilt sensor pin as input:

pinMode(tiltSensorPin, INPUT);

// Set buzzer pin as output:

pinMode(buzzerPin, OUTPUT);

Serial.begin(9600);

}

void loop()

{

int forceValue = analogRead(forceSensorPin); // Read force sensor value

int flexValue = analogRead(flexSensorPin); // Read flex sensor value

int tiltValue = digitalRead(tiltSensorPin); // Read tilt sensor value

// Check if the tilt sensor is activated or not:

if (tiltValue == HIGH)

{

// Activate the piezo buzzer with a frequency of 1000 Hz

tone(buzzerPin, 1000);

}

else

{

// Deactivate the piezo buzzer

noTone(buzzerPin);

}

delay(100);

// Print the sensor values

Serial.print("Force: ");

Serial.print(forceValue);

Serial.print(" Flex: ");

Serial.print(flexValue-990);

Serial.print(" Tilt: ");

Serial.println(tiltValue);

}

* **Explanation:**

The given code represents an Arduino project that monitors sensor values, specifically a force sensor, a flex sensor, and a tilt sensor. It also includes a piezo buzzer to provide audible feedback when the tilt sensor is activated.

* **Purpose:**

The purpose of this project is to monitor the readings from various sensors and detect tilt using a tilt sensor.

The code reads the analog values from a force sensor (connected to analog pin A0) and a flex sensor (connected to analog pin A1). It also reads the digital value from a tilt sensor (connected to digital pin 2). The force and flex sensors measure physical variables such as force and flexion, while the tilt sensor detects changes in orientation or tilt. The code sets the tilt sensor pin as an input and the buzzer pin as an output using the pinMode() function. The program then enters the main loop, where it continuously reads the sensor values using the appropriate functions (analogRead() and digitalRead()).

If the tilt sensor is activated (tiltValue == HIGH), indicating a tilt or change in orientation, the piezo buzzer is activated with a frequency of 1000 Hz using the tone() function. This audible feedback can be used to indicate a specific event or trigger an alert when a tilt is detected.

When the tilt sensor is not activated, the piezo buzzer is deactivated using the noTone() function.

The code also includes serial communication lines (Serial.begin(), Serial.print(), and Serial.println()) to display the sensor values in the Arduino IDE's serial monitor. This allows users to monitor the sensor readings and observe changes in real-time.

This project can be used in applications where monitoring physical variables and detecting tilt or orientation changes are important. For example, it can be applied in robotics, interactive installations, or security systems where the detection of tilt or physical interaction is required.