Monitoring Conditions for Crop Storage and Cultivation

Introduction:

**Problem Statement**
In many poor countries and communities around the world, the main source of income for many families is the harvest of their farms. The crops that they cultivate in these farms are used as commodities that they sell to either large wholesalers or directly to consumers in dedicated marketplaces. However, the knowledge they have in how to grow, harvest and store these crops are mainly derived from familial traditions and practices that have existed for many generations. Hence, due to a lack of knowledge of modern agricultural techniques, much of their crops end up spoiled and thus unusable for either selling or consuming. In fact in larger countries such as India, poor farmers end up losing as much as Rs 55,600 cr (about 9 billion USD) as year. This massive waste of crop has to be combated in a cheap but effective way. Most importantly, it must be accessible to poor farmers around the world.
Proposed Solution

Our solution is to create a system of sensors that can monitor crop cultivation and storage conditions for a wide variety of farm and crop types, and relay reliable information to farmers about these conditions and also notify them if there is any problem. The system will rely on a microcontroller receiving input from dedicated sensors, processing the information and relaying the information to the farmer via the internet. The sensors that we have decided to use for this solution are a light sensor, an ultrasonic rangefinder, a temperature and humidity sensor, and a moisture sensor. We also intend to display the data on a small display.

Design:

Block Diagrams

Block Diagram for the Device
The LaunchPad microcontroller serves as the central hub for all of the different sensor measurements, with each sensor feeding into microcontroller. The four individual sensors function independently of each other, but each one sends the measured data to the microcontroller. The microcontroller then converts all of the analog input data into easy-to-read digital values, which are displayed using an external computer.
Launchpad Kit with available sensors

*Code Used*
See Code Appendix

**Results:**

*Qualitative Analysis*
Each sensor properly read and measured its specific properties, with the light sensor and ultrasonic range finder being most responsive to change in the environment. Electrically, the system functioned properly with enough current and voltage applied to each sensor by the LaunchPad’s internal mechanism.

*Quantitative Analysis*
Quantitatively, each sensor had ranges of values for which they were most effective. The light sensor outputs digital values of 1 to 1023, which correspond to light intensities of less than 1 lux to greater than 100 lux. The sensor was most responsive to intensities around 35 lux, which is average room lighting intensity.

The ultrasonic sensor accurately detected objects from a distance range of 2-132 cm. Anything object placed farther than that was not detected as accurately because of the presence of other objects.
The moisture sensor possessed limited capability as it only measures the presence of water rather than the percentage of moisture content. To this end, the sensor did change in value when exposed to a wet surface vs. a dry surface, outputting a 1 and a 0, respectively, to the computer.

In a normal, room temperature environment, the temperature and humidity sensor read accurate measurements which matched the thermostat measurements of the room. The sensor measured an average of about 23 degrees Celsius and a 50% humidity rating, which is comparable to average room values.

Since we were unable to obtain a location to actually measure storage conditions for different crops, there are no testing results to determine the effectiveness of existing conditions. Instead, research was done to determine the ideal conditions for one specific crop, corn, and these values were compared to the sensors’ ranges to see if the sensors could accurately detect changes from these ideal conditions. All of the ideal conditions fell within the different sensor ranges, except the moisture sensor would prove less effective because of its inability to measure percentage of water content in soil.

**Future Improvements:**

**Next Steps**

In terms of improvements, there are many ways in which the existing solution can be modified to better solve the problem. Implementing a way to transfer the data received by the microcontroller to a remote laptop and display that information is the main improvement that can be made because remote access would allow for a greater number of storage facilities to be monitored at once.

A warning system, or some type of alert, that warns farmers which of the ideal conditions is not being met would also prove a worthwhile addition because it simplifies any maintenance of the conditions by identifying the problem early.

Testing in all kinds of different conditions for different crops would also improve the functionality of the product by proving it can work in any environment rather than just the controlled environment used during initial testing.

**Conclusion:**

Throughout this project, there were a number of obstacles faced, most of which revolved around using the LaunchPad board with its unique setup and pin mapping. Beyond the specific technical issues, the development process itself was also a point of emphasis because of the constant delays to the project timeline.

The main success of the project was that the system functioned as intended; all of the sensors were able to accurately measure the conditions of the environment, with all of the data exported to the computer in a readable format. After many trials, the pin
mapping that is specific to the LaunchPad board was identified, and we were then able to adapt the code accordingly to receive all of the information from the sensors.

Programming the LaunchPad itself was also a learning process, as each of the sensors required their own dependent files within the main code. Obtaining these dependencies and understanding how they fit into the code was challenging because of our limited programming experience. Going into the program with a basic outline of what we wanted the system to do, we found it much more difficult to develop code that would reinforce that. It took several tries and different tutorials, but we were able to program effectively to achieve our goals. Displaying the data in the best way possible was another challenging area because the 4-digit display used was limiting in the amount of information that could be displayed.

The main failure of the project was our inability to implement remote access of the data. The initial solution proposal called for remote access through the Internet, but the limited timeline of the project and the constant delays caused by our inability to decipher the LaunchPad prevented us from fully realizing the solution proposal. The display of the data was also not implemented as desired, with the different sensor values simply being cycled on the 4-digit display. Ideally, the values could be rotated at the touch of a button so each sensor could be read at a different time.

Because the project timeline was delayed repeatedly, learning how to adapt and refocus the goals of the project became more important as the different deadlines approached. The amount of weeks we spent trying to manipulate the LaunchPad to our desired modifications limited the time we could focus on the more difficult aspects of the project, namely the remote transfer of data.
Code Appendix:
#include "TM1637.h"
#include "Ultrasonic.h"
#include "DHT.h"

/* Macro Define */
define CLK 39  /* 4-Digit Display clock pin */
define DIO 38  /* 4-Digit Display data pin */
define ULTRASONIC_PIN 27  /* pin of the Ultrasonic Ranger */
define LIGHT_SENSOR 23
#define MOISTURE_PIN 25
#define THRESHOLD_VALUE 300
#define LED RED_LED
#define BLINK_LED RED_LED
#define TEMP_HUMI_PIN 24
#define BUTTON_PIN 1

/* Global Variables */
TM1637 tm1637(CLK, DIO);  /* 4-Digit Display object */
Ultrasonic ultrasonic(ULTRASONIC_PIN);  /* Ultrasonic Ranger object */
DHT dht(TEMP_HUMI_PIN, DHT22);
int distance = 0;  /* variable to store the distance to obstacles in front */
int8_t bits[4] = {0};  /* array to store the single bits of the value */
int analog_value1 = 0;
int analog_value2 = 0;
int8_t t_bits[2] = {0};
int8_t h_bits[2] = {0};
int isPressed = false;
int count = 0;
int t = 0;

/* the setup() method runs once, when the sketch starts */
void setup() {
    Serial.begin(9600);
    /* Initialize 4-Digit Display */
    tm1637.init();
    tm1637.set(BRIGHT_TYPICAL);
dht.begin();

pinMode(LED, OUTPUT);
pinMode(BLINK_LED, OUTPUT);
}

/* the loop() method runs over and over again */
void loop()
{
	/*isPressed = digitalRead(BUTTON_PIN);
if (isPressed==true){
switch (count, t, t++) {

case 1:/*
	//Ultrasonic code
distance = ultrasonic.MeasureInCentimeters();
//Serial.println("Distance");
Serial.println(distance);
memset(bits, 0, 4);
for(int i = 3; i >= 0; i--)
{

	bits[i] = distance % 10;
distance = distance / 10;
tm1637.display(i, bits[i]);
}

delay(10000);
	// break;

//case 2:
//Light sensor code
analog_value1 = analogRead(LIGHT_SENSOR); /* read the value from the sensor */
//Serial.println("Light");
Serial.println(analog_value1);
memset(bits(analogRead(LIGHT_SENSOR)), /* reset array before we use it */
for(int i = 3; i >= 0; i--)
{
	/* Convert the value to individual decimal digits for display */

bits[i] = analog_value1 % 10;
analog_value1 = analog_value1 / 10;
tm1637.display(i, bits[i]); /* display value on 4-Digit Display */
}
delay(10000); // small delay so that the number doesn't change too quickly to read
// break;

// case 3:
// Moisture sensor code
analog_value2 = analogRead(MOISTURE_PIN); /* read the value from the sensor */
//Serial.println("Moisture");
Serial.println(analogRead(MOISTURE_PIN)); /* if the value is smaller than threshold, turn on LED */
if(analog_value2 < THRESHOLD_VALUE)
{
    digitalWrite(LED, LOW);
}
else
{
    digitalWrite(LED, HIGH);
}
memset(bits, 0, 4); /* reset array before we use it */
for(int i = 3; i >= 0; i--)
{
    /* Convert the value to individual decimal digits for display */
    bits[i] = analog_value2 % 10;
analog_value2 = analog_value2 / 10;
tm1637.display(i, bits[i]); /* display on 4-Digit Display */
}
delay(10000); // small delay so that the number doesn't change too quickly to read
// break;

// case 4:
// Temp/Humidity Sensor
int _temperature = dht.readTemperature(); /* read the temperature value from the sensor */
int _humidity = dht.readHumidity(); /* read the humidity value from the sensor */
// Serial.println("Temperature");
Serial.println(_temperature);
memset(t_bits, 0, 2);  /* reset array before we use it */
memset(h_bits, 0, 2);  /* reset array before we use it */

/* 4-Digit Display [0,1] is used to display temperature */
t_bits[0] = _temperature % 10;
_temperature /= 10;
t_bits[1] = _temperature % 10;

/* 4-Digit Display [2,3] is used to display humidity */
h_bits[0] = _humidity % 10;
_humidity /= 10;
h_bits[1] = _humidity % 10;

/* show it */
tm1637.display(1, t_bits[0]);
tm1637.display(0, t_bits[1]);

tm1637.display(3, h_bits[0]);
tm1637.display(2, h_bits[1]);
delay(10000);
Serial.println(_humidity);
delay(10000);
// break;
//}