102CEM Introduction to Aerospace Electronics

Task 1

The first task was to create an onboard illumination subsystem where 3 different modes could be selected from Normal mode, Surveillance mode and Silent mode. These 3 modes utilise different components in different ways. I used 2 push buttons in this project 1 to control what mode the illumination system was on and the second one to control the pattern at which the LEDs were fading between.

Normal mode is set to be used as the drone starts up. In this mode all systems are working. The first pattern or hover pattern is 4 red LEDs (2 in the rear of the drone and 2 in the front of the drone) fading within a 500ms period these 4 red LEDs are synchronized, the second button is used to switch between patterns and in the second pattern or movement pattern only utilises the 2 rear red LEDs and adds 2 green LEDs into the mix, there is a fading pattern between the 2 colours, each colour having a fading period of 250ms. In normal mode there is also an RGB LED used as a headlight for the drone this is only activated depending on the light reading given by the LDR (Light Dependant Resistor ) component in this project, basically if it reads below a certain light level which is configurable in the code the RGB LED will light up in white which I decided is the most reasonable colour for a headlight at full brightness.

In surveillance mode neither pattern is activated thus the second button or the single colour LEDs not being utilised in this mode. The RGB LED and the LDR are still functioning as they were in normal mode where the if the lumen value is below 750 the RGB LED will turn on in a white colour.

In silent mode none of the systems are operational meaning neither the hover or movement pattern are on and the LDR and RGB LED combo are not activated either.

This is a diagram of my project:

A circuit board

Description automatically generated

Task 1 Diagram Pedro Jardim Santos (9029097)

This is the code I have written to make all this possible. In the code I have also included some descriptions of certain lines of code to help interpreting what the code means these can be seen after the double brackets.

#define LDR A0 //LDR sensor connected to analog 0 on the arduino

#define RRredLED 8 //Rear Right red LED connected to digital pin 8

#define RLredLED 7 //Rear Left red LED connected to digital pin 7

#define FRredLED 6 //Front Right red LED connected to digital pin 6

#define FLredLED 5 //Front Left red LED connected to digital pin 5

#define FRgreenLED 4 //Front Right green LED connected to digital pin 4

#define FLgreenLED 3 //Front Left green LED connected to digital pin 3

#define RRGB 22 //Red component of the RGB LED connected to digital pin 22

#define GRGB 24 //Green component of the RGB LED connected to digital pin 24

#define BRGB 26 //Blue component of the RGB LED connected to digital pin 26

//-----------fading setup-----------//

unsigned long previousMillis = 0;

unsigned long previousMillis2 = 0;

long IntervalMode1 = 6;

long IntervalMode2 = 5;

int brightness = 0;

int fadeAmount = 5;

int RedBrightness = 0;

int GreenBrightness = 255;

//----------LED Button setup----------//

#define LEDButton 9

int LEDButtonState;

int lastLEDButtonState = LOW;

int LEDMode = 0;

int oldLEDMode = 0;

boolean on = false;

//----------Mode Button setup----------//

#define ModeButton 2

int ModeButtonState;

int lastModeButtonState = LOW;

int Mode = 0;

int oldMode = 0;

//----------Debouncing----------//

unsigned long lastDebounceTime = 0;

unsigned long debounceDelay = 50;

void setup() {

//Setting the 6 different LEDs as Outputs//

pinMode(FLredLED, OUTPUT);

pinMode(FRredLED, OUTPUT);

pinMode(RLredLED, OUTPUT);

pinMode(RRredLED, OUTPUT);

pinMode(FLgreenLED, OUTPUT);

pinMode(FRgreenLED, OUTPUT);

//Setting the 3 different components of the RGB LED as Outputs//

pinMode(RRGB, OUTPUT);

pinMode(GRGB, OUTPUT);

pinMode(BRGB, OUTPUT);

//Setting the 2 buttons as Inputs//

pinMode(ModeButton, INPUT);

pinMode(LEDButton, INPUT);

//Starting the serial monitor to help with debugging//

Serial.begin(9600);

}

void loop() {

//LDR SETUP//

int val = analogRead(LDR);

Serial.println(val);

//BUTTON DEBOUNCING ROUTINE//

int ModeReading = digitalRead(ModeButton);

if (ModeReading != lastModeButtonState) {

lastDebounceTime = millis();

}

if ((millis() - lastDebounceTime) > debounceDelay) {

if (ModeReading != ModeButtonState) {

ModeButtonState = ModeReading;

if (ModeButtonState == HIGH) {

Mode = oldMode + 1;

}

}

}

lastModeButtonState = ModeReading;

//-------MODE SWITCHER-------//

switch(Mode){

case 1: //-------SURVEILLANCE MODE-------//

if (val<750){

digitalWrite(RRGB,HIGH);

digitalWrite(GRGB,HIGH);

digitalWrite(BRGB,HIGH);

}else{

digitalWrite(RRGB,LOW);

digitalWrite(GRGB,LOW);

digitalWrite(BRGB,LOW);

}

analogWrite(RLredLED, LOW);

analogWrite(RRredLED, LOW);

analogWrite(FLredLED, LOW);

analogWrite(FRredLED, LOW);

analogWrite(FLgreenLED, LOW);

analogWrite(FRgreenLED, LOW);

oldMode = Mode;

break;

case 2: //-------SILENT MODE-------//

analogWrite(RLredLED, LOW);

analogWrite(RRredLED, LOW);

analogWrite(FLredLED, LOW);

analogWrite(FRredLED, LOW);

analogWrite(FLgreenLED, LOW);

analogWrite(FRgreenLED, LOW);

digitalWrite(RRGB,LOW);

digitalWrite(GRGB,LOW);

digitalWrite(BRGB,LOW);

oldMode = Mode;

break;

default: //-------NORMAL MODE-------//

if (val<750){

digitalWrite(RRGB,HIGH);

digitalWrite(GRGB,HIGH);

digitalWrite(BRGB,HIGH);

}else{

digitalWrite(RRGB,LOW);

digitalWrite(GRGB,LOW);

digitalWrite(BRGB,LOW);

}

LEDButtonState = digitalRead(LEDButton); // latch switch code for the button on the breadboard to prevent bouncing and bad connections

if (LEDButtonState == HIGH){

delay(50);

LEDButtonState = digitalRead(LEDButton);

if(LEDButtonState == LOW){

if(on==true){

on=false;

} else{

on=true;

}

}

}

if(on == false){

unsigned long currentMillis = millis(); // grab current time

analogWrite(FLredLED, brightness);// set the brightness of ledPin:

analogWrite(FRredLED, brightness);

analogWrite(RLredLED, brightness);

analogWrite(RRredLED, brightness);

digitalWrite(FLgreenLED,LOW);

digitalWrite(FRgreenLED,LOW);

if (currentMillis - previousMillis >= IntervalMode1){

brightness = brightness + fadeAmount; // change the brightness for next time through the loop:

previousMillis = millis();

}

if (brightness <= 0 )

{ // reverse the direction of the fading at the ends of the fade:

brightness = 0;

fadeAmount = -fadeAmount;

}

if (brightness >=255 )

{ // reverse the direction of the fading at the ends of the fade:

brightness = 255;

fadeAmount = -fadeAmount;

}

}

else if(on == true){

// set the brightness of pin 9:

analogWrite(RLredLED, RedBrightness);

analogWrite(RRredLED, RedBrightness);

analogWrite(FLredLED, LOW);

analogWrite(FRredLED, LOW);

// set the brightness of pin 10:

analogWrite(FLgreenLED, GreenBrightness);

analogWrite(FRgreenLED, GreenBrightness);

// change the brightness for next time through the loop:

RedBrightness = RedBrightness + fadeAmount;

GreenBrightness = GreenBrightness - fadeAmount;

// reverse the direction of the fading at the ends of the fade:

// and blink the internal LED on pin 13 at the fade end point

if (RedBrightness == 0 || RedBrightness == 255) {

fadeAmount = -fadeAmount ;

}

// wait for 30 milliseconds to see the dimming effect

delay(8); // physiscal button acts up without this line and wont bridge the connection correctly

}

oldMode = 0;

break;

}

}

Task 2

In task 2 the project was to make a diagnostics test for the drone’s subsystems before it could be turned on. There’s 4 tests that need to be taken and these can be done in any order, the IMU test, SV test, Motor test and WiFi test. These tests are in a menu displayed in a 16x2 LCD display and the menu is controlled by joystick and the tests are selected using the button on the joystick.

When the Arduino is started up the LCD is blank and nothing is displayed, by pushing a button on the breadboard the LCD will display the message “Testing mode ON” this means the menu is now accessible and the tests can be carried out by scrolling down you can select what test you want and while scrolling you will see different messages displayed as “IMU test” and “press to start” respectively for each test. The tests are then started by pushing the button on the joystick and the message “press to start” is changed for “testing”. The testing period lasts for 5 seconds for every test and it accompanied by an RGB LED blinking at a period of 200ms in a blue colour. If nothing else is pressed the test will be successful and the RGB LED will turn green and the system returns to the menu to select the next test, there is a second button on the breadboard used to simulate an error with the test, while the test is going on if the button is held down the test will be failed and along with a buzzer tone the RGB LED will turn red, this will still put you back to the menu where you can retry the test.

Once you are done with the test you can choose to leave by scrolling to the bottom of the menu where the message “Testing mode” and “press to exit” will be displayed, if the joystick button is pressed at this moment you will leave the menu and the LCD will go back to blank.

This is a diagram of my project:

Task 2 Diagram
Pedro Jardim Santos (9029097)

Task 2 Diagram Pedro Jardim Santos (9029097)

This is the code used for this project as with task 1 I have added a few descriptions to make the code easier to understand. In this task I also used a lot more functions to facilitate the code you can see the functions when their names are after the void prefix eg. “void menu” these exclude “void setup” and “void loop” as these are standard with every Arduino sketch.

#include <Wire.h>

#include <LiquidCrystal.h>

LiquidCrystal LCD(7, 8, 9, 10, 11, 12); // the LCD is connected to digital pins 7 through to 12

#define buzzer 5 // the buzzer is connected to digital pin 5

#define Rled 22 // red portion of RGB led connected to digital pin 22

#define Gled 24 // green portion of RGB led connected to digital pin 24

#define Bled 26 // blue portion of RGB led connected to digital pin 26

//---------------Joystick Setup---------------//

#define xPin A1 // X axis component on the joystick is connected to analog pin 1

#define yPin A0 // Y axis component on the joystick is connected to analog pin 0

#define kPin 13 // Button on the joystick is connected to digital pin 13

int Yaxis;

int Xaxis;

int joyRead;

int joyPos;

int lastJoyPos;

long lastDebounceTime = 0;

long debounceDelay = 70;

bool refresh;

bool PQCP;

//---------------LCD Button Setup---------------//

#define Button 4

int lastButtonState = LOW;

int ButtonState;

int LCDmode = 0;

int oldLCDmode = 0;

//---------------Error Button Setup---------------//

#define Error 3

int lastErrorState = LOW;

int ErrorState;

int lastexitKState = LOW;

int exitKState;

//--------------Debouncing Routine--------------//

int lastkState = 0;

int kState = HIGH;

int kMode = 0;

int oldkMode = 0;

int Mode = 0;

int oldMode = 0;

void setup() {

//Setting the joystick pins to Inputs//

pinMode(xPin, INPUT);

pinMode(yPin, INPUT);

pinMode(kPin, INPUT\_PULLUP);

//Setting the 3 different components of the RGB LED as Outputs//

pinMode(Rled, OUTPUT);

pinMode(Gled, OUTPUT);

pinMode(Bled, OUTPUT);

//Buzzer is set as an Output//

pinMode(buzzer, OUTPUT);

//Letting the arduino know that there is 16 columns and 2 rows//

LCD.begin(16, 2);

//Initiating the serial monitor to help with debugging when necessary//

Serial.begin(9600);

}

void loop() {

Buttons();

ONOFF();

}

void ONOFF() {

switch (LCDmode) {

case 1:

LCD.display();

controlJoystick();

Menu();

oldLCDmode = LCDmode;

break;

default:

LCD.noDisplay();

oldLCDmode = 0;

break;

}

}

void Menu() {

switch (Yaxis) {

case 1:

IMUTest();

break;

case 2:

SVTest();

break;

case 3:

MotorTest();

break;

case 4:

WIFITest();

break;

case 5:

Exit();

break;

default:

TestingON();

break;

}

}

void LEDBlinking() {

digitalWrite(Bled, HIGH);

delay(100);

digitalWrite(Bled, LOW);

delay(100);

Serial.println("Testing...");

}

void Buttons() {

int ButtonReading = digitalRead(Button);

if (ButtonReading != lastButtonState) {

lastDebounceTime = millis();

}

if ((millis() - lastDebounceTime) > debounceDelay) {

if (ButtonReading != ButtonState) {

ButtonState = ButtonReading;

if (ButtonState == HIGH) {

LCDmode = oldLCDmode + 1;

}

}

}

lastButtonState = ButtonReading;

}

void ErrorCheck() {

int ErrorReading = digitalRead(Error);

if (ErrorReading == HIGH) {

tone(buzzer, 3000);

digitalWrite(Rled, HIGH);

delay(1000);

digitalWrite(Rled, LOW);

noTone(buzzer);

Serial.println("ERROR");

}

else if (ErrorReading == LOW) {

digitalWrite(Gled, HIGH);

delay(1000);

digitalWrite(Gled, LOW);

Serial.println("SUCCESS");

}

}

void IMUTest() {

int k = digitalRead(kPin);

if (k != lastkState) {

lastDebounceTime = millis();

}

if ((millis() - lastDebounceTime) > debounceDelay) {

if (k != kState) {

kState = k;

if (kState == LOW) {

kMode = oldkMode + 1;

}

}

}

lastkState = k;

switch (kMode) {

case 1:

if (refresh) {

LCD.clear();

refresh = 0;

}

LCD.clear();

LCD.setCursor(0, 0);

LCD.print("IMU Test");

delay(10);

LCD.setCursor(0, 1);

LCD.print("Testing");

int counter;

for (int x = 0; x <= 25; x++) {

LEDBlinking();

}

ErrorCheck();

kMode = 0;

default:

if (refresh) {

LCD.clear();

refresh = 0;

}

LCD.setCursor(0, 0);

LCD.print("IMU Test");

delay(10);

LCD.setCursor(0, 1);

LCD.print("press to start");

oldkMode = 0;

break;

}

}

void SVTest() {

int k = digitalRead(kPin);

if (k != lastkState) {

lastDebounceTime = millis();

}

if ((millis() - lastDebounceTime) > debounceDelay) {

if (k != kState) {

kState = k;

if (kState == LOW) {

kMode = oldkMode + 1;

}

}

}

lastkState = k;

switch (kMode) {

case 1:

if (refresh) {

LCD.clear();

refresh = 0;

}

LCD.clear();

LCD.setCursor(0, 0);

LCD.print("SV Test");

delay(10);

LCD.setCursor(0, 1);

LCD.print("Testing");

int counter;

for (int x = 0; x <= 25; x++) {

LEDBlinking();

}

ErrorCheck();

kMode = 0;

default:

if (refresh) {

LCD.clear();

refresh = 0;

}

LCD.setCursor(0, 0);

LCD.print("SV Test");

delay(10);

LCD.setCursor(0, 1);

LCD.print("press to start");

oldkMode = 0;

break;

}

}

void MotorTest() {

int k = digitalRead(kPin);

if (k != lastkState) {

lastDebounceTime = millis();

}

if ((millis() - lastDebounceTime) > debounceDelay) {

if (k != kState) {

kState = k;

if (kState == LOW) {

kMode = oldkMode + 1;

}

}

}

lastkState = k;

switch (kMode) {

case 1:

if (refresh) {

LCD.clear();

refresh = 0;

}

LCD.clear();

LCD.setCursor(0, 0);

LCD.print("Motor Test");

delay(10);

LCD.setCursor(0, 1);

LCD.print("Testing");

int counter;

for (int x = 0; x <= 25; x++) {

LEDBlinking();

}

ErrorCheck();

kMode = 0;

default:

if (refresh) {

LCD.clear();

refresh = 0;

}

LCD.setCursor(0, 0);

LCD.print("Motor Test");

delay(10);

LCD.setCursor(0, 1);

LCD.print("press to start");

oldkMode = 0;

break;

}

}

void WIFITest() {

int k = digitalRead(kPin);

if (k != lastkState) {

lastDebounceTime = millis();

}

if ((millis() - lastDebounceTime) > debounceDelay) {

if (k != kState) {

kState = k;

if (kState == LOW) {

kMode = oldkMode + 1;

}

}

}

lastkState = k;

switch (kMode) {

case 1:

if (refresh) {

LCD.clear();

refresh = 0;

}

LCD.clear();

LCD.setCursor(0, 0);

LCD.print("WIFI Test");

delay(10);

LCD.setCursor(0, 1);

LCD.print("Testing");

int counter;

for (int x = 0; x <= 25; x++) {

LEDBlinking();

}

ErrorCheck();

kMode = 0;

default:

if (refresh) {

LCD.clear();

refresh = 0;

}

LCD.setCursor(0, 0);

LCD.print("WIFI Test");

delay(10);

LCD.setCursor(0, 1);

LCD.print("press to start");

oldkMode = 0;

break;

}

}

void Exit() {

if (refresh) {

LCD.clear();

refresh = 0;

}

LCD.setCursor(0, 0);

LCD.print("Testing Mode");

delay(10);

LCD.setCursor(0, 1);

LCD.print("press to exit");

int exitK = digitalRead(kPin);

if (exitK != lastexitKState) {

lastDebounceTime = millis();

}

if ((millis() - lastDebounceTime) > debounceDelay) {

if (exitK != exitKState) {

exitKState = exitK;

if (exitKState == LOW) {

LCD.clear();

Yaxis = 0;

LCDmode = 0;

}

}

}

lastexitKState = exitK;

}

void TestingON() {

if (refresh) {

LCD.clear();

refresh = 0;

}

LCD.setCursor(0, 0);

LCD.print("Testing Mode");

delay(10);

LCD.setCursor(0, 1);

LCD.print("ON");

}

void controlJoystick() {

leeJoystick();

if (PQCP) {

PQCP = 0;

LCD.noBlink();

if (Yaxis < 8 && joyPos == 3) {

Yaxis++; //Down

refresh = 1;

Xaxis = 0;

}

if (Yaxis > 0 && joyPos == 4) {

Yaxis--; //Up

Xaxis = 0;

refresh = 1;

}

if (Xaxis < 3 && joyPos == 1) {

Xaxis++; //Right

refresh = 1;

}

if (Xaxis > 0 && joyPos == 2) {

Xaxis--; //Left

refresh = 1;

}

}

}

int leeJoystick() {

int x = analogRead(xPin);

int y = analogRead(yPin);

if (x > 900) {

joyRead = 1; //x+

} else if (x < 100) {

joyRead = 2; //x-

} else if (y > 900) {

joyRead = 3; //y+

} else if (y < 100) {

joyRead = 4; //y-

} else {

joyRead = 0;

}

//----------------Deboucing for the Joystick----------------//

if (joyRead != lastJoyPos) {

lastDebounceTime = millis();

}

if (((millis() - lastDebounceTime) > debounceDelay) && (joyRead != joyPos)) {

joyPos = joyRead;

if (!PQCP) {

PQCP = 1;

}

}

if (((millis() - lastDebounceTime) > (5 \* debounceDelay)) && (joyPos == 3 || joyPos == 4)) {

joyPos = joyRead; //repeat time only for UP/DOWN

if (!PQCP) {

PQCP = 1;

}

}

lastJoyPos = joyRead;

}

I am including 2 YouTube links to unlisted videos where the demonstration of both tasks is done:

Task 1:

<https://youtu.be/rDt56XlYz1c>

Task 2:

<https://youtu.be/ch61C6PtVCQ>

I am also including a GitHub link for a private repository where you can access my original Arduino sketches in the native .ino file extension this link is only for convenience if needed and not for anything else.

<https://github.com/PedroJardimSantos53/102CEM-coursework>