**ДОДАТОК А**

**Лістинг коду світлофора**

001: // Simple sketch for traffic lights with pedestrian cross by button press

002: #include <Servo.h> // include library for working with servo

003: Servo myservo; // create servo object to control a servo

004:

005: // constants for cars led pins

006: const int carsRedLedPin = 6;

007: const int carsYellowLedPin = 5;

008: const int carsGreenLedPin = 4;

009:

010: // constants for pedestrian led pins

011: const int pedestrianRedLedPin = 2;

012: const int pedestrianGreenLedPin = 12;

013:

014: const int buttonPin = 10; // const for button pin

015: const int zoomerPin = 3; // const for zoomer pin

016: const int servoPin = 11; // const for servo pin

017: const int distanceSensorTriggerPin = 7; // const for distance sensor trigger pin

018: const int distanceSensorEchoPin = 8; // const for distance sensor echo pin

019: const int photoresistorPin = A0; // const for photoresistor pin

020: const int temperatureSensorPin = A5; // const for tmp36 pin

021: const int variant = 60; // my variant number

022: const int redGreenDuration = 60; // work duration time for red and green led

023: const int dangerDistance = 60; // value from distant sensor considered to be danger distance

024:

025: bool buttonState = false; // variable to read button state

026: bool enablePedestrian; // flag variable for pedestrian loop

027: bool enableDangerAlert; // flag variable for danger alert loop

028: unsigned int dangerCheckTime; // variable to check how much time pass since checking distance

029:

030: float getTemperature() {

031: float sensorInput = analogRead(temperatureSensorPin); //read the analog sensor and store it

032: float temp = (float)sensorInput / 1024; // find percentage of input reading

033: temp = temp \* 5; // multiply by 5V to get voltage

034: temp = temp - 0.5; // subtract the offset

035: temp = temp \* 100; // convert to degrees

036: return temp;

037: }

038:

039: int getDinstance() {

040: int duration, distance;

041:

042: digitalWrite(distanceSensorTriggerPin, LOW); // set trigger pin to low for more accurate results

043: delayMicroseconds(2);

044:

045: digitalWrite(distanceSensorTriggerPin, HIGH); // set trigget pin to high

046:

047: delayMicroseconds(10); // wait 10 Œºs

048: digitalWrite(distanceSensorTriggerPin, LOW);

049:

050: duration = pulseIn(distanceSensorEchoPin, HIGH); // get high signal duration on the echo pin

051: distance = duration / 58; // copmuting distance

052: return distance;

053: }

054:

055: void setZoomer() {

056: int photoresistorValue = analogRead(photoresistorPin); // get value from photoresistor

057: int toneValue = map(photoresistorValue, 0, 1023, 50, 20000); // map it to a frequency range

058:

059: noTone(zoomerPin); // disable previous tone

060: tone(zoomerPin, toneValue); // enable new tone

061: }

062:

063: void setZoomerDanger() {

064: enableDangerAlert = true; // set danger alert flag to true

065:

066: noTone(zoomerPin); // disable previous tone

067: tone(zoomerPin, (50 + variant \* 200)); // enable new tone

068: }

069:

070: void openBarrier(int workTime) {

071: for (int pos = 0; pos <= 90; pos += 1) { // goes from 0 degrees to 90 degrees

072: // in steps of 1 degree

073: myservo.write(pos); // tell servo to go to position in variable 'pos'

074: delay(workTime / 90); // waits worktime / 90 ms for the servo to reach the position

075: }

076: }

077:

078: void closeBarrier(int workTime) {

079: for (int pos = 90; pos >= 0; pos -= 1) { // goes from 90 degrees to 0 degrees

080: // in steps of 1 degree

081: myservo.write(pos); // tell servo to go to position in variable 'pos'

082: delay(workTime / 90); // waits worktime / 90 ms for the servo to reach the position

083: }

084: }

085:

086: void setup() {

087: Serial.begin(9600); //Start the Serial Port at 9600 baud

088: // setting leds pins pinMode to output

089: pinMode(carsGreenLedPin, OUTPUT);

090: pinMode(carsYellowLedPin, OUTPUT);

091: pinMode(carsRedLedPin, OUTPUT);

092: pinMode(pedestrianGreenLedPin, OUTPUT);

093: pinMode(pedestrianRedLedPin, OUTPUT);

094:

095: pinMode(distanceSensorTriggerPin, OUTPUT); // setting distace sensor trigger pinMode to output

096: pinMode(buttonPin, INPUT); // setting button pin pinMode to input

097: pinMode(temperatureSensorPin, INPUT); // setting tmp36 pin pinMode to input

098:

099: myservo.attach(servoPin); // attaches the servo on servoPin to the servo object

100:

101: digitalWrite(pedestrianRedLedPin, HIGH); // enabling pedestrian red led cause it should be enabled due to normal loop

102: setZoomer(); // enabling zoomer with red pedestriab led

103: closeBarrier(1350);

104: }

105:

106: void loop() {

107: setZoomer(); // re enabling zoomer to update tone frequency

108:

109: digitalWrite(carsGreenLedPin, HIGH); // enable green cars led

110:

111: dangerCheckTime = millis(); // save time when checking distant loop begins

112: while(millis() < dangerCheckTime + redGreenDuration \* 1000) {

113: // the loop will be broken after (redGreenDuration)

114: if(getDinstance() <= dangerDistance) {

115: enableDangerAlert = true;

116: setZoomerDanger();

117: }

118: }

119:

120: digitalWrite(carsGreenLedPin, LOW); // disable green cars led

121:

122: digitalWrite(carsYellowLedPin, HIGH); // enable yellow led

123: delay(2000);

124:

125: buttonState = digitalRead(buttonPin); // read button state

126: if(buttonState == HIGH) {

127: enablePedestrian = true; // if button pressed set enablePedestrian flag to true

128: openBarrier(3000);

129: } else {

130: delay(3000);

131: }

132:

133: digitalWrite(carsYellowLedPin, LOW); // disable yellow led

134:

135: digitalWrite(carsRedLedPin, HIGH); // enable red cars led

136: // check if danger alert is enabled

137: if (enableDangerAlert) {

138: noTone(zoomerPin); // disable danger alert zoomer

139: enableDangerAlert = false; // set danger alert flag to false

140: setZoomer(); // enable zoomer as it must work due to pedestrian red led

141: }

142:

143: // checking if pedestrian lights must be used

144: if(enablePedestrian) {

145: // enabling red cars led, disabling pedestrian red, and anebling pedestrian green

146: digitalWrite(pedestrianRedLedPin, LOW);

147: digitalWrite(pedestrianGreenLedPin, HIGH);

148: noTone(zoomerPin); // disable pedestrian red light zoomer

149: delay(redGreenDuration \* 1000);

150:

151: digitalWrite(pedestrianGreenLedPin, LOW); // disabling pedestrian green

152: digitalWrite(pedestrianRedLedPin, HIGH); // enabling pedestrian red

153: setZoomer(); // enabling pedestrian red light zoomer

154: digitalWrite(carsRedLedPin, LOW); // disabling cars red

155: digitalWrite(carsYellowLedPin, HIGH); // anabling cars yellow led

156: closeBarrier(5000); // close barier

157: enablePedestrian = false; // disabling pedestrian flag

158: } else {

159: // enabling cars red led

160: delay(redGreenDuration \* 1000);

161: digitalWrite(carsRedLedPin, LOW); // disabling cars red led

162: digitalWrite(carsYellowLedPin, HIGH); // anabling cars yellow led

163: delay(5000); // yellow light daley

164: }

165:

166: digitalWrite(carsYellowLedPin, LOW); // disabling cars yellow led

167:

168: Serial.println("Current temperature: ");

169: Serial.println(getTemperature());

170:

171: Serial.println("Current light level: ");

172: Serial.println(analogRead(photoresistorPin));

173:

174: Serial.println("Current distance: ");

175: Serial.println(getDinstance());

176: }

177: