Autor: Cocioran Ștefan

Automatic bus door closer

Introducere

Mecanismul are rolul de a realiza închiderea ușilor unui mijloc de transport în comun (ex. autobuz) în condiții de siguranță. Atunci când un călător dorește să urce în autobuz, este posibil ca ușile să se închidă brusc și să îi prindă haina, geanta sau chiar să îl rănească. Pentru a evita astfel de situații, mecanismul folosit va detecta dacă se află persoane în proximitatea ușilor mijlocului de transport în comun înainte de a le închide.

Descriere generală

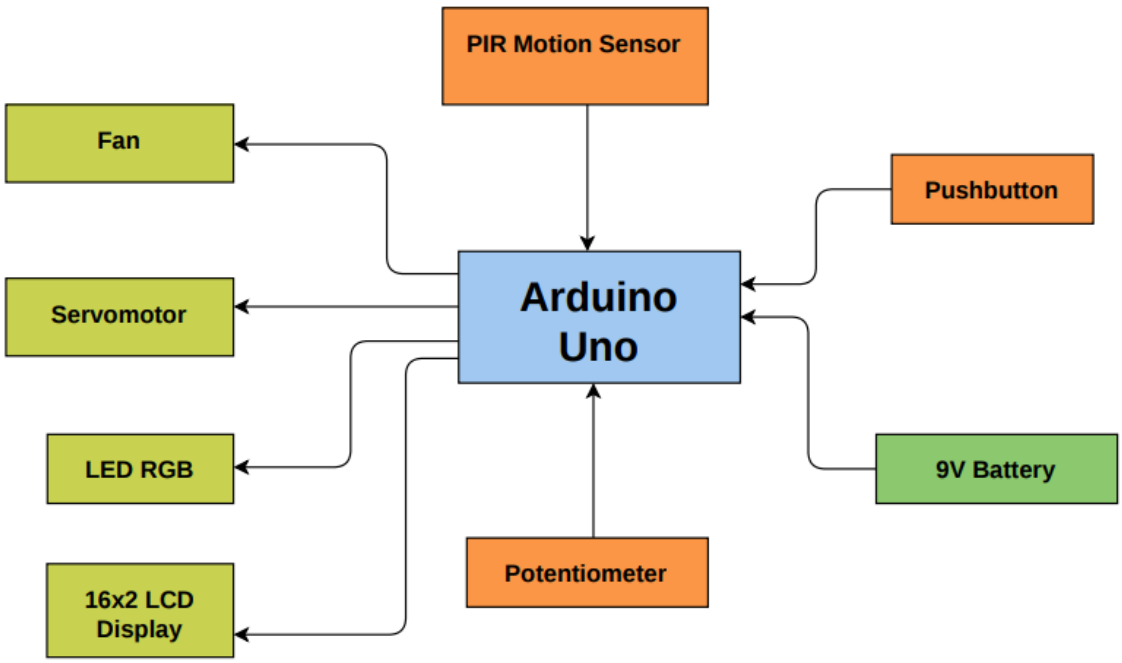
Se va lega un ventilator la placa Arduino Uno și un display LCD. Ventilatorul va porni atunci când vehiculul se deplasează și reprezinta motorul acestuia, a cărui viteză va fi reglată prin intermediul unui potențiometru și afișată pe LCD (se face o conversie din RPM în km/h).

Pentru a închide/deschide ușile autobuzului se apasă un buton și este acționat servomotorul. Când ușile sunt închise un LED RGB se va aprinde roșu, iar când sunt deschise acesta va fi aprins verde. De asemenea, starea ușilor va fi afișată pe LCD.

Se amplasează senzorul de mișcare PIR la ușa autobuzului pentru a vedea dacă se află călători în apropiere care vor să urce/coboare din autobuz, starea acestuia va fi activă pentru 3 secunde de la ultima mișcare înregistrată.

Dacă se apasă butonul pentru închiderea ușilor și senzorul indică faptul că se află un călător în apropiere, ușile vor rămâne deschise. De asemenea, pentru că se măsoară și viteză autobuzului, am adăugat o funcționalitate că ușile să se închidă automat după ce este depășită o anumită viteză (10km/h).

Schema bloc



Hardware Design

Listă componente:

- Arduino UNO
- Buton
- LED RGB
- PIR senzor mișcare
- Potențiometru
- Ventilator
- Servomotor
- Display LCD 16×2
- Rezistor 220Ω x 2 pentru LED RGB
- Rezistor 1kΩ pentru contrast LCD
- Rezistor 10kΩ pentru buton
- Dioda redresoare 1N4007
- Tranzistor 2N2222

Schema electrică

Software Design

Dezvoltarea codului s-a realizat folosind Arduino IDE.

Funcționare

- Inițial ușile autobuzului sunt închise.
- Viteza acestuia este reglată cu ajutorul potențiometrului și convertite în km/h folosind map.
- În funcție de viteză ventilatorul va porni sau se va opri (când este sub 3km/h).
- În bucla principală se afișează în permanență pe LCD viteza autobozului și starea ușilor acestuia cu ajutorul bibliotecii LiquidCrystal, iar cu biblioteca Servo se realizează mișcarea ușilor prin intermediul servomotorului.
- Dacă viteza de 10km/h este depășită, ușile se vor închide singure fără a apăsa pe buton.
- Atunci când este acționat butonul ușile se vor deschide (doar dacă autobuzul stă pe loc) sau închide.
- Se citesc în permanență date de la senzorul de mișcare atunci când se dorește închiderea ușilor.
- Dacă a fost detectată mișcare în proximitatea ușilor în momentul respectiv, acestea se vor deschide la loc și se va afișa un mesaj cu ***Someone at the door!***.

automatic_bus_door_closer.c

```
#include <LiquidCrystal.h>
#include <Servo.h>
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// Variables
Servo servo;
int speed = 0;                // current vehicle speed
int pos = 0;                  // current servomotor position
int ledState = LOW;     // the current state of the output pin
int buttonState;          // the current reading from the input pin
int lastButtonState = LOW; // the previous reading from the input pin
float startTime;          // when motion is detected and  doors fully
open
unsigned long lastDebounceTime = 0; // the last time the output pin was
toggled
unsigned long debounceDelay =
    10; // the debounce time; increase if the output flickers
bool motionDetected = false;

// Constants
const int closed_pos = 0, open_pos = 130, servo_delay = 15;
const int max_speed = 10, start_moving_speed = 3;
const float printMotionTime = 2500;
const unsigned int calibrationTime = 60;
```

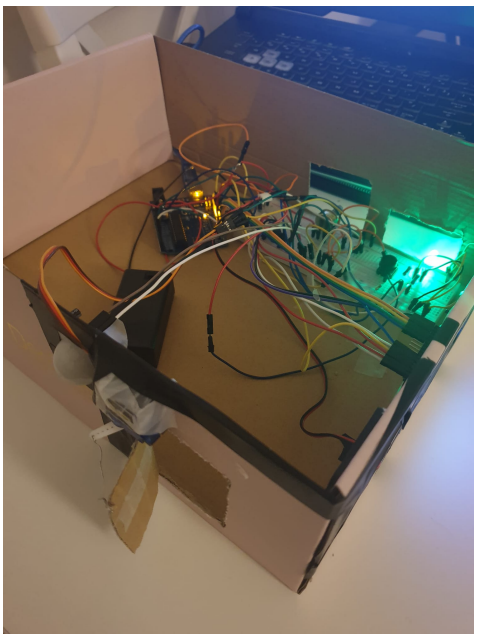
```
// Pins
const int sensorPin = 6;
const int buttonPin = 7;
const int ledPinRed = 8;
const int servoPin = 9;
const int motorPin = 10;
const int ledPinGreen = 13;

void setup()
{
        // Set up the LCD's number of columns and rows:
        lcd.begin(16, 2);
        pinMode(sensorPin, INPUT);
        pinMode(buttonPin, INPUT);
        pinMode(ledPinGreen, OUTPUT);
        pinMode(ledPinRed, OUTPUT);
        pinMode(motorPin, OUTPUT);

        Serial.begin(9600);
        servo.attach(servoPin, 500, 2500);
        servo.write(closed_pos);

        for (int i = 0; i < calibrationTime; i++) {
                lcd.setCursor(0, 0);
                lcd.print("Calibrating");
                lcd.setCursor(5, 1);
                lcd.print("sensor...");
                delay(1000);
        }
        lcd.clear();

        // Make sure vehicle is not moving when the simulation begins
        speed = map(analogRead(A0), 0, 999, 0, 115);
        speed -= speed;
}

void print_movement()
{
        lcd.setCursor(0, 0);
        lcd.print("Someone at the");
        lcd.setCursor(5, 1);
        lcd.print("door!");
}

void print_speed()
{
        // Conversion from RMP to km/h
        speed = map(analogRead(A0), 0, 999, 0, 115);

        if (speed >= start_moving_speed)
```

```
                digitalWrite(motorPin, HIGH);
        else
                digitalWrite(motorPin, LOW);

        lcd.setCursor(0, 0);
        lcd.print("Speed: ");
        lcd.print(speed);
        lcd.print(" km/h   ");
}

void print_door_state()
{
        lcd.setCursor(0, 1);
        lcd.print("Door: ");
        if (!ledState)
                lcd.print("Closed");
        else
                lcd.print("Opened");
}

void open_door()
{
        for (int i = pos; i <= open_pos; i++, pos++) {
                if (speed >= max_speed) {
                        ledState = LOW;
                        close_door();
                        break;
                }

                // Tell servo to go to position in variable 'pos'
                servo.write(pos);
                delay(servo_delay);

                if (motionDetected) {
                        print_movement();
                } else {
                        print_speed();
                }
        }
        startTime = millis();
}

void close_door()
{
        for (int i = pos; i >= closed_pos; i--, pos--) {
                if (digitalRead(sensorPin) == HIGH && speed < max_speed)
{
                        ledState = HIGH;
                        motionDetected = true;
                        lcd.clear();
                        open_door();
```

```
                        break;
                }

                // Tell servo to go to position in variable 'pos'
                servo.write(pos);
                delay(servo_delay);
                print_speed();
        }
}

void loop()
{
        // Set the LED:
        if (ledState) {
                digitalWrite(ledPinRed, LOW);
                digitalWrite(ledPinGreen, HIGH);
        } else {
                digitalWrite(ledPinGreen, LOW);
                digitalWrite(ledPinRed, HIGH);
        }

        if (motionDetected) {
                if (millis() - startTime < printMotionTime) {
                        print_movement();
                        return;
                }

                lcd.clear();
                motionDetected = false;
                Serial.println(pos);
                return;
        }

        print_speed();
        print_door_state();

        if (ledState != LOW && speed >= max_speed) {
                ledState = LOW;
                close_door();
                return;
        }

        // Read the state of the switch into a local variable
        int reading = digitalRead(buttonPin);

        // If the switch changed, due to noise or pressing:
        if (reading != lastButtonState) {
                // Reset the debouncing timer
                lastDebounceTime = millis();
        }
```

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

                // If the button state has changed:
                if (reading != buttonState) {
                        buttonState = reading;

                        // Only toggle the LED if the new button state
is HIGH
                        if (buttonState == HIGH)
                                if (!ledState && speed <
start_moving_speed) {
                                        ledState = HIGH;
                                        open_door();
                                }

                                else {
                                        ledState = LOW;
                                        close_door();
                                }
                }
        }

        // Save the reading, it'll be the lastButtonState:
        lastButtonState = reading;
}
```

Rezultate Obţinute



Concluzii

Realizând acest proiect am observat că lucrurile în practică stau mult mai diferit față de cum mă așteptam. Inițial am implementat circuitul în Tinkercad și a mers foarte bine, însă în realitate piesele (în special motoarele) consuma foarte mult curent și perturbau funcționarea la parametrii normali a celorlalte componente (de ex. LCD-ul se stingea când funcționa motorul DC, motiv pentru care l-am înlocuit cu un ventilator alimentat separat de o baterie de 9V). Întâmpinând astfel de probleme și încercând să le rezolv, pot spune că am înteles mai bine anumite noțiunile de electronică și sunt mulțumit de rezultatul final.

Download

Arhivă: automatic_bus_door_closer.zip

Jurnal

- **13/05/2022** - Publicare Introducere + Descriere generală
- **27/05/2022** - Schema electrică + Software design
- **31/05/2022** - Modificare circuit, înlocuire motor DC cu ventilator și adăugare baterie 9V

Bibliografie/Resurse

ATmega328P Datasheet

PIR Motion Sensor Datasheet

Export to PDF