Design with microprocessors project

Traffic Lights

A picture containing text, gauge

Description automatically generated

Lungoci Luca

Rus Iulia-Maria

30434

Prof : Attila Fuzes

Contents :

1) Overview

2) Components

3) Schematic

4) Explanation of the flow

5) Code

Project overview

The main idea of this project is to simulate a traffic intersection lighting system. There is a semaphore for pedestrians, whose road intersects the car road, and there are two other semaphores for cars that work together, meaning that they are not both green at the same time.

It could obviously be used to model the semaphore schematic for a real intersection that resembles the one that we have created:

A picture containing text

Description automatically generated

This intersection is not very complex, but it does require attention if you don’t want any accidents, which is why when the semaphore on top is green, the others cannot be also green. The same principle applies to each and everyone of them.

Besides its mentioned possible application, it was interesting to implement as an experiment, in order to work with Arduinos and learn about serial communication and hardware components.

Components

* Arduino UNO – we chose this over Arduino mega both because it was way cheaper and easy to find, and also had all the ports and functionalities that we needed to implement our project.
* LEDs – they were easy to find and also cheap. They were also the first thing we thought of in regards to the semaphore’s lights.
* A button
* Resistances – used for safely handling the LEDs.
* A motherboard – in order for us to hook up all the wires together.

Project Schematic

The project schematic was made in tinkercad.

Diagram

Description automatically generated

Program Flow

* Board on the right

The pedestrian semaphore works in a rather simple manner:

It stays red until someone presses the button. When someone presses the button hooked up at pin 2 it causes an external interrupt. An interrupt service routine is called, and that modifies a global variable that decides the colour of the semaphore . If that variable, that also has a suggestive name: ‘changedByInterr’ is true, then it means that the button was pressed and we have to turn on the green light. After the green light stays on for 5 seconds, it is switched and the semaphore is red again for pedestrians.

As for serial connection, we used the I2C protocol where the board on which we work with this semaphore plays the *master role*. We connected the A4, A5 and GND , and used functions from the Wire library in order to send a message when the colour of the pedestrian semaphore changes.

* Board on the left

Considering the serial connection for the two boards, the board on the left is the slave. It is responsible for controlling the lights for the two car roads and receives messages from the other board. The messages are sent in order to turn the lights red for the cars because the pedestrians are crossing and another message is send when the normal execution can be resumed.

Execution flow:

* The program executes periodically (at each 4 seconds) the two functions responsible for switching the green and red lights, through the “every” function of “Timer.h”. Two provide two types of functionality (one when no pedestrians wish to cross the road and one when there are any) the program uses a “received” variable. When “received” is ‘1’ it means that people wish to cross the road so, the two functions are still called periodically, but every attempt to switch a green light on and a red light off is stopped by the condition “ if( received != '1') ”. When a message is received through the wire, the function handling the interrupt is “button\_pressed” which turns both red light on, green lights off and updated the “received” variable. Regarding the switching of the semaphores, a global Boolean variable is used to make one pair of semaphores showing green and the other showing red. It modified by the function “switch\_light\_green” where can also be found some delays for the blinking functionality.

The Code

* **Board on left**

#include "Timer.h"

#include <Wire.h>

char received;

Timer t1, t2;

int RED1 = 3, GREEN1 = 4, RED2 = 5, GREEN2 = 6;

bool toggle = false;

bool toggle\_red = true;

bool keep\_red = false;

void setup() {

  Wire.begin(9);

  Wire.onReceive(button\_pressed);

  pinMode(RED1, OUTPUT);

  pinMode(GREEN1, OUTPUT);

  pinMode(RED2, OUTPUT);

  pinMode(GREEN2, OUTPUT);

  digitalWrite(GREEN2, HIGH);

  digitalWrite(RED1, HIGH);

  t1.every(4000, switch\_light\_red);

  t2.every(4000, switch\_light\_green);

}

void loop() {

  t1.update();

  t2.update();

}

void button\_pressed(int bytes) {

  received = Wire.read();

  if (received == '1') {

    digitalWrite(RED1, HIGH);

    digitalWrite(RED2, HIGH);

    digitalWrite(GREEN1, LOW);

    digitalWrite(GREEN2, LOW);

  }

  Serial.println(received);

}

void switch\_light\_red() {

  int light\_on = RED1, light\_off = RED2;

  if (toggle\_red) {

    light\_on = RED2;

    light\_off = RED1;

    toggle\_red = false;

  } else

    toggle\_red = true;

  digitalWrite(light\_on, HIGH);

  if (received != '1')

    digitalWrite(light\_off, LOW);

}

void switch\_light\_green() {

  //true -> GREEN1     false -> GREEN2

  int to\_switch = GREEN1;

  if (toggle) {

    toggle = false;

  } else {

    to\_switch = GREEN2;

    toggle = true;

  }

  digitalWrite(GREEN1, LOW);

  digitalWrite(GREEN2, LOW);

  long current\_millis = millis();

  for (int i = 0; i < 2; i++) {

    while (millis() - current\_millis < 250)

      ;

    current\_millis = millis();

    if (received != '1')

      digitalWrite(to\_switch, HIGH);

    while (millis() - current\_millis < 250)

      ;

    current\_millis = millis();

    digitalWrite(to\_switch, LOW);

  }

  if (received == '1')

    return;

  if (toggle)

    digitalWrite(GREEN1, HIGH);

* else
* digitalWrite(GREEN2, HIGH);
* }
* **Board on right**

#include <Wire.h>

volatile bool changedByInterr = false;

volatile bool notSetAlready = true;

long innerTimer = 0;

int b1;

long secondInnerTimer = 0;

void setup() {

  Serial.begin(9600);

  Wire.begin();

  pinMode(2, INPUT\_PULLUP);

  pinMode(5, OUTPUT);

  pinMode(4, OUTPUT);

  attachInterrupt(digitalPinToInterrupt(2), TurnGreen, FALLING);

  digitalWrite(4, LOW);

  digitalWrite(5, HIGH);

}

void loop() {

  normalSemaphore();

}

void normalSemaphore(void) {

  if (changedByInterr) {

    digitalWrite(5, LOW);

    digitalWrite(4, HIGH);

    changedByInterr = false;

    Wire.beginTransmission(9);

    Wire.write('1');

    Wire.endTransmission();

    delay(5000);

    Wire.beginTransmission(9);

    Wire.write('0');

    Wire.endTransmission();

  } else {

    digitalWrite(5, HIGH);

    digitalWrite(4, LOW);

  }

}

void TurnGreen(void) {

  Serial.println("button ");

  if (digitalRead(4) == LOW) {

    changedByInterr = true;

  }

}