Proiect SMP

**Definiție**:

De realizat un sistem de control fără fir al unui utilizator casnic.

**Descriere:**

Am înlocuit un utilizator casnic cu o altă plăcută Arduino ce primește de la sistemul de control comanda de aprindere/stingere a ledului ce se regăsește pe plăcuță. Clientul va transmite către sistemul de comandă un semnal pentru a-i comunica statusul clientului (ledului) și afișează pe ecran acest status.

**Hardware utilizat:**

* *Plăcuță Arduino Uno*
* *Receptor RF 433 MHZ* -> Primește statusul ledului/dispozitivului
* *Emitor RF 433 MHz* -> Transmite comanda către dispozitiv
* *Led* -> semnalizarea apăsării butonului pentru transmiterea comenzii
* *Butoane* -> transmite impuls către arduino pentru a transmite comanda către client
* *Modul RTC* -> Măsurarea timpului pentru auto-aprinderea/stingerea ledului/dispozitivului
* *Display led* -> Afișarea stării led-urilor

**Implementare:**

**

*/\*<-- Libraries -->\*/*

*#include "Functions.h"*

*bool isOk\_1 = false;*

*bool isOk\_2 = false;*

*void setup() {*

*// Leds*

*pinMode(led\_sent, OUTPUT);*

*pinMode(led\_recieved, OUTPUT);*

*// Reciever*

*Serial.begin(9600);*

*mySwitchRx.enableReceive(0); // Receiver on interrupt 0 => that is pin #2*

*// Transmitter*

*mySwitchTx.enableTransmit(7); // Using Pin #7 --> Transmitter*

*mySwitchTx.setProtocol(protocol);*

*mySwitchTx.setPulseLength(pulseLength);*

*// Display*

*lcd.begin(16, 2);*

*lcd.clear();*

*// Real Time Clockx*

*if (! rtc.begin()) {*

*Serial.print("Couldn't find RTC");*

*while (1);*

*}*

*if (! rtc.isrunning()) {*

*Serial.print("RTC is NOT running!");*

*}*

*rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));*

*// READY TO GO!*

*change\_time = rtc.now();*

*change\_time\_2 = rtc.now();*

*lcd.setCursor(0, 0);*

*lcd.print("1 - " + String(button1\_action));*

*lcd.setCursor(0, 1);*

*lcd.print("2 - " + String(button1\_action\_2));*

*}*

*void loop() {*

*DateTime now = rtc.now();*

*isOk\_1 = false;*

*isOk\_2 = false;*

*//Reciever*

*if (mySwitchRx.available()) {*

*Serial.println("Value: " + String(mySwitchRx.getReceivedValue()));*

*Serial.println("Delay: " + String(mySwitchRx.getReceivedDelay()));*

*Serial.println("Protocol: " + String(mySwitchRx.getReceivedProtocol()));*

*if (mySwitchRx.getReceivedProtocol() == 2) {*

*if (mySwitchRx.getReceivedValue() == 12165000) {*

*isOk\_1 = true;*

*}*

*else if (mySwitchRx.getReceivedValue() == 12165001) {*

*isOk\_1 = false;*

*}*

*else if (mySwitchRx.getReceivedValue() == 22165001) {*

*isOk\_2 = true;*

*}*

*else if (mySwitchRx.getReceivedValue() == 22165001) {*

*isOk\_2 = false;*

*}*

*else{*

*isOk\_1 = false;*

*isOk\_2 = false;*

*}*

*}*

*displayRecievedMessage(lcd, mySwitchRx.getReceivedValue());*

*change\_time = rtc.now();*

*mySwitchRx.resetAvailable();*

*}*

*// Transmitter*

*/\**

*The message will be send when the user will press the button (ON/OFF) or automatically after 5 seconds since last send.*

*\*/*

*button1\_status = digitalRead(buttonPin);*

*if (button1\_status == HIGH || now.unixtime() - change\_time.unixtime() > 10 ) {*

*button1\_action = ! button1\_action;*

*if (button1\_action) {*

*last\_code\_on = sendCode(mySwitchTx, last\_code\_on, button1\_action);*

*}*

*else {*

*last\_code\_off = sendCode(mySwitchTx, last\_code\_off, button1\_action);*

*}*

*digitalWrite(led\_sent, (button1\_action ? HIGH : LOW));*

*change\_time = rtc.now();*

*}*

*// Button 2*

*button1\_status\_2 = digitalRead(buttonPin\_2);*

*if (button1\_status\_2 == HIGH || now.unixtime() - change\_time\_2.unixtime() > 30) {*

*button1\_action\_2 = ! button1\_action\_2;*

*if (button1\_action\_2) {*

*last\_code\_on\_2 = sendCode2(mySwitchTx, last\_code\_on\_2, button1\_action\_2);*

*}*

*else {*

*last\_code\_off\_2 = sendCode2(mySwitchTx, last\_code\_off\_2, button1\_action\_2);*

*}*

*change\_time\_2 = rtc.now();*

*}*

*lcd.setCursor(0, 0);*

*lcd.print("1 - " + String(isOk\_1));*

*lcd.setCursor(0, 1);*

*lcd.print("2 - " + String(isOk\_1));*

*}*

**Fișierul header pentru funcții:**

*#include "Variables.h"*

*/\*<-- FUNCTIONS -->\*/*

*// Auxiliar functions*

*int sizeOf(unsigned long \*input) {*

*return sizeof(\*input) / sizeof(input[0]);*

*}*

*int sizeOf(int \*input) {*

*return sizeof(\*input) / sizeof(input[0]);*

*}*

*// Get the type of code: on (true) or off (false)*

*bool getTypeOfCode(unsigned long code) {*

*for (int i = 0; i < sizeOf(code\_on); i++) {*

*if (code == code\_on[i]) {*

*return true;*

*}*

*}*

*return false;*

*}*

*// Get code index*

*int getCodeIndex (unsigned long code) {*

*unsigned long\* codeType = ( getTypeOfCode(code) ? code\_on : code\_off);*

*for (int i = 0; i < sizeOf(codeType); i++) {*

*if (code == codeType[i]) {*

*return i;*

*}*

*}*

*return -1;*

*}*

*// Send code*

*unsigned long sendCode (RCSwitch switchTX, unsigned long last\_code, bool signalType) {*

*int index = 0;*

*if (signalType) {*

*for (int k = 0; k < sizeof(code\_on) / sizeof(code\_on[0]); k++) {*

*if (last\_code = code\_on[k]) {*

*index = k;*

*}*

*}*

*last\_code = index != 3 ? code\_on[++index] : code\_on[0];*

*}*

*else {*

*for (int k = 0; k < sizeof(code\_off) / sizeof(code\_off[0]); k++) {*

*if (last\_code = code\_off[k]) {*

*index = k;*

*}*

*}*

*last\_code = index != 3 ? code\_off[++index] : code\_off[0];*

*}*

*switchTX.send(last\_code, 24);*

*Serial.println("Protocol: " + String(protocol) + "; Delay: " + String(pulseLength) + "; Code: " + String(last\_code));*

*return last\_code;*

*}*

*unsigned long sendCode2 (RCSwitch switchTX, unsigned long last\_code, bool signalType) {*

*int index = 0;*

*if (signalType) {*

*for (int k = 0; k < sizeof(code\_on\_2) / sizeof(code\_on\_2[0]); k++) {*

*if (last\_code = code\_on\_2[k]) {*

*index = k;*

*}*

*}*

*last\_code = index != 3 ? code\_on\_2[++index] : code\_on\_2[0];*

*}*

*else {*

*for (int k = 0; k < sizeof(code\_off\_2) / sizeof(code\_off\_2[0]); k++) {*

*if (last\_code = code\_off\_2[k]) {*

*index = k;*

*}*

*}*

*last\_code = index != 3 ? code\_off\_2[++index] : code\_off\_2[0];*

*}*

*switchTX.send(last\_code, 24);*

*Serial.println("Protocol: " + String(protocol) + "; Delay: " + String(pulseLength) + "; Code: " + String(last\_code));*

*return last\_code;*

*}*

*// Display Fuctions*

*void displayRecievedMessage(LiquidCrystal LCD, unsigned long sentCode) {*

*LCD.setCursor(0, 1);*

*LCD.print("R.: " + String(sentCode));*

*}*

*void displaySentMessage(LiquidCrystal LCD, unsigned long sentCode) {*

*LCD.clear();*

*LCD.setCursor(0, 0);*

*LCD.print("S.: " + String(sentCode));*

*}*

*void displaySocketStatus(LiquidCrystal LCD, int buttonId, bool buttonState, int numberOfPresses) {*

*LCD.setCursor(0, 0);*

*LCD.print(String(buttonId) + "." + (buttonState == true ? "ON " : "OFF") + " " + String(numberOfPresses));*

*}*

*void displayTime(LiquidCrystal LCD, DateTime now) {*

*// lcd.setCursor(0, 1);*

*lcd.print(now.hour());*

*lcd.print(':');*

*lcd.print(now.minute());*

*lcd.print(':');*

*lcd.print(now.second());*

*lcd.print(" ");*

*// lcd.setCursor(0, 0);*

*// lcd.print(daysOfTheWeek[now.dayOfTheWeek()]);*

*// lcd.print(" ,");*

*// lcd.print(now.day());*

*// lcd.print('/');*

*// lcd.print(now.month());*

*// lcd.print('/');*

*// lcd.print(now.year());*

*}*

***Fișierul header cu variabilele:***

*#include <RCSwitch.h> //--> Rx + Tx*

*#include <LiquidCrystal.h> //Display*

*#include "RTClib.h"*

*/\*<-- VARIABLES -->\*/*

*// Real Time Clock Module*

*RTC\_DS1307 rtc;*

*// RF 433 MHz*

*// - Reciever*

*RCSwitch mySwitchRx = RCSwitch();*

*// - Transmitter*

*RCSwitch mySwitchTx = RCSwitch();*

*//Display*

*const int rs = 8, en = 9, d4 = 5, d5 = 4, d6 = 3, d7 = 6;*

*LiquidCrystal lcd(rs, en, d4, d5, d6, d7);*

*// Button*

*const int buttonPin = 10;*

*bool button1\_status = false;*

*bool button1\_action = false;*

*const int buttonPin\_2 = 11;*

*bool button1\_status\_2 = false;*

*bool button1\_action\_2 = false;*

*// LEDS*

*int const led\_sent = 13;*

*int const led\_recieved = 10;*

*// Auxiliar variables*

*DateTime change\_time;*

*unsigned long last\_code\_on = 12165804;*

*unsigned long last\_code\_off = 11807932;*

*DateTime change\_time\_2;*

*unsigned long last\_code\_on\_2 = 22165804;*

*unsigned long last\_code\_off\_2 = 21807932;*

*unsigned long const code\_on[] = {12165804, 11696236, 12518172, 11567196};*

*unsigned long const code\_off[] = {11807932, 12470652, 12319532, 11982220};*

*unsigned long const code\_on\_2[] = {22165804, 21696236, 22518172, 21567196};*

*unsigned long const code\_off\_2[] = {21807932, 22470652, 22319532, 21982220};*

*int const protocol = 3;*

*int const pulseLength = 101;*

*char daysOfTheWeek[7][12] = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"};*