PROIECT SMP

Sistem de control fără fir pentru controlul utilizatorilor casnici

Cătălin-Răzvan Barbălată
332AA

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:


Functions.h


Variables.h


SMP_Project.ino

```
/*<-- 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
  /*
```

Fișierul header pentru funcții:

```
#include "Variables.h"

/*<-- FUNCTIONS -->*/
```

```
    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));
 }



// 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"};
```