Projektni izveštaj

Luka Krivačević RM 96/15

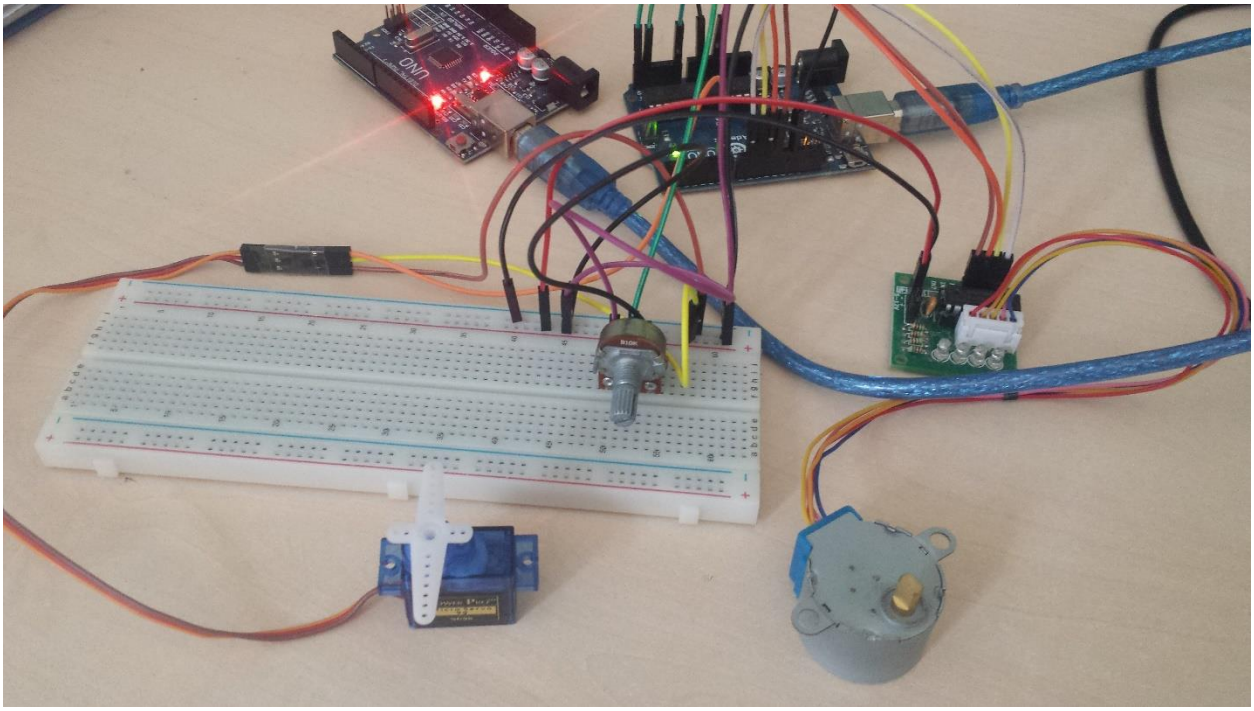
Postavka projektnog zadatka

Konstruisati sistem koji se sastoji od dva arduino medjusobno povezanih (master-slave). Na masteru povezane sledeće komponente: servo motor, step motor (sa ULN2003 drajverom), potenciometar.

Pri startovanju sistema, bira se konfiguracija (default ili custom). Zatim se bira protocol (I2C ili SPI), s tim sto SPI nije implementiran jos.

Sistem ima 3 mašine stanja: idle (sistem je u stanju mirovanja), servo (upravlja se servo motorom), step (upravlja se step motorom). Upravljanje se vrši preko serial veze uz mogućnost i upravljanja preko potenciometra.

Slave prima od mastera vrednosti koje su zadate motorima i ispisuje ih na serial monitor.



Opis hardvera

Master Arduino UNO:

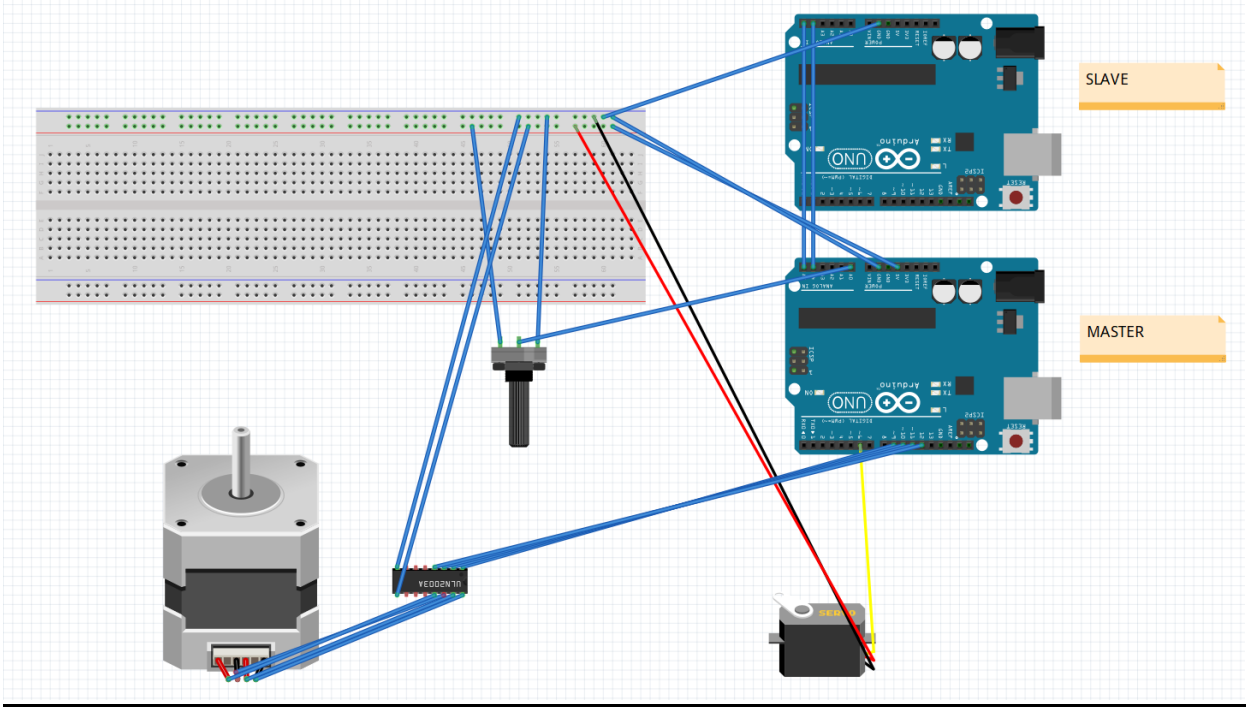
Analog pin A5, A4 – I2C veza sa slave arduinom

Analog pin A0 – Potenciometar

Digital pin 6 – Servo motor

Digital pin 9, 10, 11, 12 – Step motor (ULN2003 drajver)

Slave Arduino Uno



Algoritam

Pri startovanju sistema nudi se opcija default konfiguracije ili custom konfiguracije, kao i protokola I2C ili SPI (SPI nije implementiran). Nakon odabira, sistem ulazi u idle mašinsko stanje određeno boolean promenljivom machineState.

Unosom “servo” preko seriala, ulazi se u stanje kontrolisanja servo motora (machineState = 1) preko serial veze. Unete vrednosti se kreću od 0-180 za položaj servo motora. U slučaju veće/manje vrednosti,

automatski se ažuriraju vrednosti na 0/180. Takodje postoji opcija kontrolisanja motora preko potenciometra unosom “potenc” preko seriala. Master automatski šalje unete vrednosti slejvu.

Unosom “step” preko seriala, ulazi se u stanje kontrolisanja step motora (machineState = 2) preko serial veze. Unete vrednosti se kreću od 2-500 u zavisnosti od željene brzine motora. U slučaju veće/manje vrednosti, automatski se ažuriraju vrednosti na 500/2. Takodje postoji opcija kontrolisanja motora preko potenciometra unosom “potenc” preko seriala. Master automatski šalje unete vrednosti slejvu.

Pri startovanju slejva se takođe bira protokol I2C ili SPI (SPI nije implementiran). Pri dobijanju vrednosti od mastera, ispisuje ih na serial monitor.

CODE

MASTER

```
#include <Wire.h>

#include <SPI.h>

#include <Servo.h>


const int cs = 5;


#define STEPPER_PIN_1 9

#define STEPPER_PIN_2 10

#define STEPPER_PIN_3 11

#define STEPPER_PIN_4 12

int step_number = 0;

boolean changeDirection = false;

int servoPin = 6;

int potencPin = 0;


Servo servo1;


boolean potencUsed = false;
```

```
boolean configurationChosen = false;
boolean protocolChosen = false;
boolean I2C = true; //if it's set false, SPI is selected

String serialString;  //serialStringRead();
String protocol;

int machineState = 0; //0 - idle, 1 - servo, 2 - step



void setup() {

  pinMode(STEPPER_PIN_1, OUTPUT);
  pinMode(STEPPER_PIN_2, OUTPUT);
  pinMode(STEPPER_PIN_3, OUTPUT);
  pinMode(STEPPER_PIN_4, OUTPUT);

  Serial.begin(9600);
  //CONFIGURATION
  if (!configurationChosen) {
    Serial.println("1 - Default configuration");
    Serial.println("2 - Custom configuration");
    setupStringRead();
    if (serialString == "1\n") {
      configurationChosen = true;
      serialString = "";
      Serial.println("Default configuration selected.");
      Serial.println("Servo motor - pin 9");
```

```
    servo1.attach(servoPin);
  } else if (serialString == "2\n") {
    configurationChosen = true;
    serialString = "";
    Serial.println("Custom configuration selected.");
    //custom pins
  } else {
    Serial.println("Unknown command.");
    delay(20);
    configurationChosen = false;
    serialString = "";
    setup();
  }
}

//PROTOCOL
if (!protocolChosen) {
  Serial.println("Choose protocol (I2C or SPI):");
  setupStringRead();
  if (serialString == "I2C\n") {
    serialString = "";
    protocolChosen = true;
    Serial.println("I2C protocol selected.");
    I2C = true;
    Wire.begin();
  } else if (serialString == "SPI\n") {
    serialString = "";
    protocolChosen = true;
    Serial.println("SPI protocol selected.");
```

```
    I2C = false;
    pinMode(cs, OUTPUT);
    SPI.begin ();
    SPI.setClockDivider(SPI_CLOCK_DIV8);
    SPI.setBitORder(MSBFIRT);
   } else {
    Serial.println("Unknown command");
    delay(20);
    serialString = "";
    protocolChosen = false;
    setup();
   }
  }
}

byte x = 0;
int servoPosition;
int stepSpeed = 2;

void loop() {

  switch (machineState) {
   case 0: //idle
    //Serial.println("Idle state");
    if (Serial.available()) {
     String tmpString = Serial.readString();
     checkStateChange(tmpString);
    }
    break;
```

```
case 1: //servo -> (0-180) SERVO MODE
  //Serial.println("Servo state");
  if (Serial.available()) {
    String tmpString = Serial.readString();
    checkStateChange(tmpString);  //check state change 1st
    if (machineState != 1) {
      serialString = "";
      break;
    }
    if (tmpString == "potenc\n") { //check if potenc is used 2nd
      potencUsed = !potencUsed; //revert boolean for on/off
    }
    servoPosition = tmpString.toInt();  //if not, it's from serial
  }

  if (potencUsed) { //if it feeds via potentiometer
    servoPosition = analogRead(potencPin) / 5; //BAGUJE GLUPI POTENCIOMETAR
    Serial.println(servoPosition);
  }
  if (servoPosition > 180) {
    servoPosition = 180;
  } else if (servoPosition < 0) {
    servoPosition = 0;
  }
  servo1.write(servoPosition);

  //send to slave
  if (I2C) {  //ako je I2C
    Wire.beginTransmission(10); // transmit to device #8
```

```
      Wire.write("Servo position is: "); // sends five bytes
      Wire.write(servoPosition); // sends one byte
      Wire.endTransmission(); // stop transmitting
    } else {  //ako je SPI
      // enable Slave Select
      digitalWrite(SS, LOW);    // SS is pin 10
      // send servo position
//      for (const char * p = "Hello, world!\n" ; c = *p; p++)
//        SPI.transfer (c);

      // disable Slave Select
      digitalWrite(SS, HIGH);
    }

    break;
   case 2: //STEP MODE
    //Serial.println("Step state");
    if (Serial.available()) {
      String tmpString = Serial.readString();
      checkStateChange(tmpString);  //check state change
      if (machineState != 2) {
        serialString = "";
        break;
      }
      if (tmpString == "potenc\n") { //check if potenc is used 2nd
        potencUsed = !potencUsed; //revert boolean for on/off
      }
      if (tmpString == "change direction\n") { //if to change direction
        changeDirection = !changeDirection; //revert boolean for changing direction
```

```
  } else {
    stepSpeed = tmpString.toInt();  //if not, it's from serial
  }
}

if (potencUsed) { //if it feeds via potentiometer
  stepSpeed = (analogRead(potencPin) / 6) - 30; //BAGUJE GLUPI POTENCIOMETAR
  Serial.println(stepSpeed);
}
if (stepSpeed > 500) {
  stepSpeed = 500;
} else if (stepSpeed < 2) {
  stepSpeed = 2;
}
OneStep(changeDirection);
delay(stepSpeed);
Serial.println(stepSpeed);

//send to slave
if (I2C) {  //ako je I2C
  Wire.beginTransmission(10); // transmit to device #8
  Wire.write("Step speed is: "); // sends five bytes
  Wire.write(stepSpeed); // sends one byte
  Wire.endTransmission(); // stop transmitting
} else {  //ako je SPI

}

break;
```

```
    default:  //idle
      //Serial.println("Idle state");
      if (Serial.available()) {
        String tmpString = Serial.readString();
        checkStateChange(tmpString);
      }
      break;
  }

  //  if (I2C) {  //ako je I2C
  //    Wire.beginTransmission(10); // transmit to device #8
  //    // Wire.write("x is "); // sends five bytes
  //    Wire.write(x); // sends one byte
  //    Wire.endTransmission(); // stop transmitting
  //
  //    x++;
  //    delay(500);
  //  } else {  //ako je SPI
  //
  //  }
}

void setupStringRead() {
  while (!Serial.available()) {
    ;
  }
  while (Serial.available()) {
    serialString = Serial.readString(); // read the incoming data as string
  }
```

```
}

void serialStringRead() {
  while (Serial.available()) {
    serialString = Serial.readString(); // read the incoming data as string
  }
}

void checkStateChange(String myString) {
  if (myString == "idle\n") {
    Serial.println("Idle state");
    machineState = 0;
  } else if (myString == "servo\n") {
    Serial.println("Servo state");
    machineState = 1;
  } else if (myString == "step\n") {
    Serial.println("Step state");
    machineState = 2;
  } else {

  }
}

void OneStep(bool dir) {
  if (dir) {
    switch (step_number) {
      case 0:
        digitalWrite(STEPPER_PIN_1, HIGH);
        digitalWrite(STEPPER_PIN_2, LOW);
```

```
      digitalWrite(STEPPER_PIN_3, LOW);
      digitalWrite(STEPPER_PIN_4, LOW);
      break;
    case 1:
      digitalWrite(STEPPER_PIN_1, LOW);
      digitalWrite(STEPPER_PIN_2, HIGH);
      digitalWrite(STEPPER_PIN_3, LOW);
      digitalWrite(STEPPER_PIN_4, LOW);
      break;
    case 2:
      digitalWrite(STEPPER_PIN_1, LOW);
      digitalWrite(STEPPER_PIN_2, LOW);
      digitalWrite(STEPPER_PIN_3, HIGH);
      digitalWrite(STEPPER_PIN_4, LOW);
      break;
    case 3:
      digitalWrite(STEPPER_PIN_1, LOW);
      digitalWrite(STEPPER_PIN_2, LOW);
      digitalWrite(STEPPER_PIN_3, LOW);
      digitalWrite(STEPPER_PIN_4, HIGH);
      break;
  }
} else {
  switch (step_number) {
    case 0:
      digitalWrite(STEPPER_PIN_1, LOW);
      digitalWrite(STEPPER_PIN_2, LOW);
      digitalWrite(STEPPER_PIN_3, LOW);
      digitalWrite(STEPPER_PIN_4, HIGH);
```

```
      break;
    case 1:
      digitalWrite(STEPPER_PIN_1, LOW);
      digitalWrite(STEPPER_PIN_2, LOW);
      digitalWrite(STEPPER_PIN_3, HIGH);
      digitalWrite(STEPPER_PIN_4, LOW);
      break;
    case 2:
      digitalWrite(STEPPER_PIN_1, LOW);
      digitalWrite(STEPPER_PIN_2, HIGH);
      digitalWrite(STEPPER_PIN_3, LOW);
      digitalWrite(STEPPER_PIN_4, LOW);
      break;
    case 3:
      digitalWrite(STEPPER_PIN_1, HIGH);
      digitalWrite(STEPPER_PIN_2, LOW);
      digitalWrite(STEPPER_PIN_3, LOW);
      digitalWrite(STEPPER_PIN_4, LOW);


   }
  }
  step_number++;
  if (step_number > 3) {
    step_number = 0;
  }
}
```

Slave

```
#include <Wire.h>
#include <SPI.h>

boolean protocolChosen = false;
String serialString;
boolean I2C = true; //if it's set false, SPI is selected

void setup() {
  Serial.begin(9600);

  //PROTOCOL
  if (!protocolChosen) {
    Serial.println("Choose protocol (I2C or SPI):");
    serialStringRead();
    if (serialString == "I2C\n") {
      serialString = "";
      protocolChosen = true;
      Serial.println("I2C protocol selected.");
      I2C = true;
      Wire.begin(10); // join i2c bus with address #10
      Wire.onReceive(receiveEvent); // register event
    } else if (serialString == "SPI\n") {
      serialString = "";
      protocolChosen = true;
      Serial.println("SPI protocol selected.");
```

```
    I2C = false;
    //OTVARANJE SPI KONEKCIJE
   } else {
    Serial.println("Unknown command");
    delay(20);
    serialString = "";
    protocolChosen = false;
    setup();
   }
 }
}

void loop() {
}
void receiveEvent(int howMany) {
 while (1 < Wire.available()) { // loop through all but the last
  char c = Wire.read(); // receive byte as a character
  Serial.print(c); // print the character
 }
 int x = Wire.read(); // receive byte as an integer
 Serial.println(x); // print the integer
}

void serialStringRead() {
 while (!Serial.available()) {
  ;
 }
 while (Serial.available()) {
  serialString = Serial.readString(); // read the incoming data as string
```

```
  }
}
```