Tehnički fakultet Rijeka

Računalom podržana mjerenja

Seminarski rad

Izradili: Deni Klen, Ani Perušić, Mihael Petranović, Mateo Srića

Mentor: prof. dr. sc. Saša Vlahinić

Rijeka, ožujak 2021

Sadržaj

Uvod

Tema seminarskog rada je izrada grafičkog sučelja u Python-u koje će prikazivati podatke o temperaturi i vlažnosti pomoću senzora spojenog na Arduino mikročip, kao i izrada data loggera, uređaja koji prikuplja podatke sa senzora i sprema ih za kasnije korištenje.

Razvojna okruženja i sklopovlje

Arduino platforma

Mikročipska ploča korištena u izradi projekta je Arduino Micro. Ona je jedna od Arduinovih mikročipovskih ploča koje su napravljene za početnike; jednostavna je za upotrebu te pisanje koda, a služi kao dobra polazna točka za upoznavanje i učenje elektronike. Zasnonavana je na Atmegi32U4 i razvijena zajedno s Adafruitom. Ima 20 digitalnih ulazno/izlaznih pinova (od kojih se 7 mogu koristiti kao PWM izlazi, a 12 kao analogni ulazi), kristalni oscilator od 16 MHz, mikro USB vezu, ICSP zaglavlje i gumb za resetiranje. Sadrži sve potrebno za podršku mikrokontrolera i jednostavno se spaja na računalo uz pomoć mikro USB kabela. Ugrađena USB komunikacija uklanja potrebu za sekundarnim procesorom. To omogućuje mikročipskoj ploči da se na povezanom računalu prikaže kao miš i tipkovnica, uz virtualni (CDC) serijski / COM priključak.

Programsku podršku za mikročip pisali smo u Arduino IDE-u. To je softver otvorenog koda koji olakšava pisanje programske podrške i prijenos na bilo koju Arduino ploču. Aktivni razvoj se odvija na GitHub-u.

Python

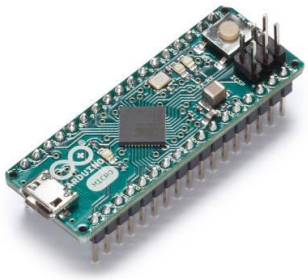
Za izradu grafičkog sučelja koristili smo programski jezik Python. Python je programski jezik visoke razine i opće namjene koji omogućava brzi rad i učinkovitu integraciju sustava. Njegove jezične konstrukcije i objektno orijentirani pristup imaju kao cilj pomoći programerima da napišu jasan i logičan kod bez obzira o veličini projekta. Python se dinamički upisuje i prikuplja smeće, stoga programer na mora misliti o alokaciji i brisanju memorije, podržava više paradigmi programiranja, uključujući strukturirano (posebno proceduralno), objektno

orijentirano i funkcionalno programiranje. Python se često opisuje kao jezik "s uključenim baterijama" zbog svoje sveobuhvatne standardne biblioteke koja uvelike olakšava programiranje.

Senzor

Senzor korišten za prikupljanje podataka o temperaturi i vlažnosti je DHT11. U njemu se nalazi senzor temperature – mali termistor zalemljen na pločicu te senzor vlage – mala tiskana pločica koja je nadolemljena na osnovnu. Što je više vlage u zraku, više vlage dolazi i na same vodove te je otpor među njima manji (ili počinje postojati kada se pojavljuje voda — to je razlog zašto senzor očitava vrijednosti od 20%, tek onda može očitati nekakav otpor). Osim navedene dvije komponente još nalazimo jedan integrirani krug koji analzira ulaze od prethodno navedenih senzora i komunicira s Arduinom.

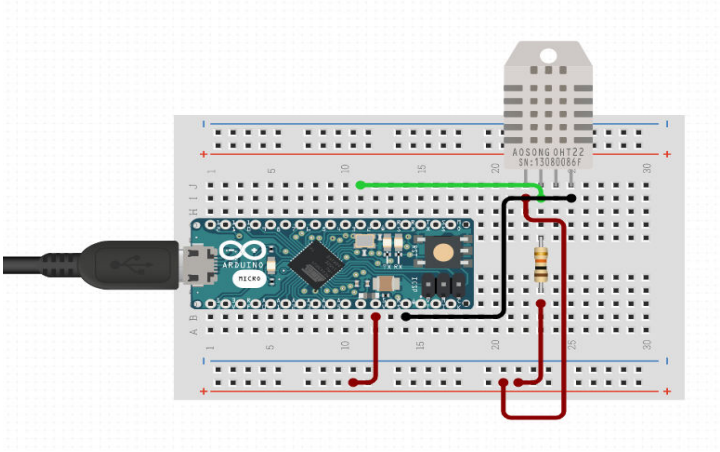
Slika 1. VMA311 DHT11 senzor



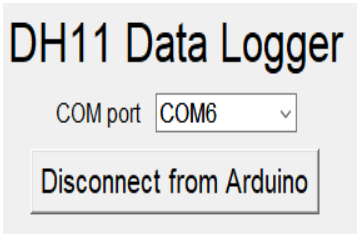
Slika 2. Arduino Micro

Opis rada

Prije otvaranja programa, potrebno je spojiti Arduino pločicu na računalo i senzor na Arduino pločicu. Senzor se na Arduino spaja na 5V, uzemljenje i analogni ulaz A0. Nakon pravilnog spajanja se može otvoriti program. Potrebno je odabrati port na koji je Arduino spojen nakon čega se može pokrenuti konekcija između računala i Arduina gumbom *Connect to Arduino*. U svakom trenutku se može ista konekcija zaustaviti gumbom *Disconnect from Arduino*.

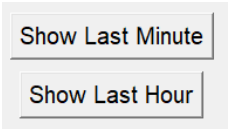


Slika 3. DHT22/11 senzor spojen na Arduino Micro

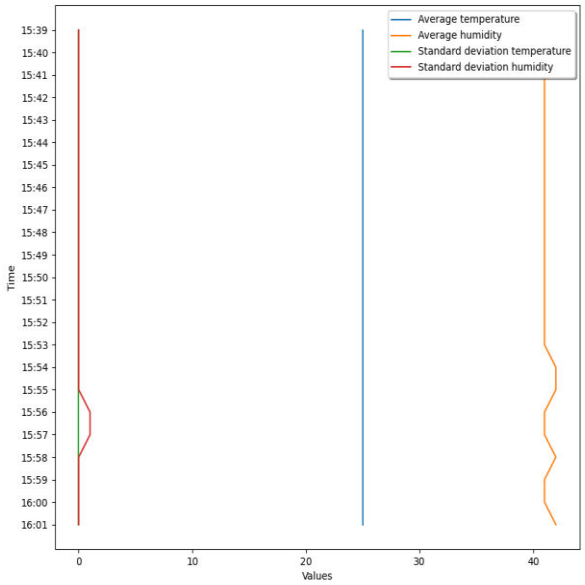


Slika 4. Prikaz zaglavlja prozora

Sučelje se sastoji od dva gumba – *Show Last Minute* i *Show Last Hour* koji pritiskom pokazuju grafove i podatkovne vrijednosti u tablici.



Slika 5. Gumbi za prikaz podataka



Slika 6. Graf vrijednosti za protekli sat

Hour Statistics

Time	Average temperature	Average humidity	Standard deviation temperature	Standard deviation humidity
16:01	25.0	42.0	0.0	0.0
16:00	25.0	41.0	0.0	0.0
15:59	25.0	41.0	0.0	0.0
15:58	25.0	42.0	0.0	0.0
15:57	25.0	41.0	0.0	1.0
15:56	25.0	41.0	0.0	1.0
15:55	25.0	42.0	0.0	0.0
15:54	25.0	42.0	0.0	0.0
15:53	25.0	41.0	0.0	0.0
15:52	25.0	41.0	0.0	0.0
15:51	25.0	41.0	0.0	0.0
15:50	25.0	41.0	0.0	0.0
15:49	25.0	41.0	0.0	0.0
15:48	25.0	41.0	0.0	0.0
15:47	25.0	41.0	0.0	0.0
15:46	25.0	41.0	0.0	0.0
15:45	25.0	41.0	0.0	0.0

Slika 7. Sirovi prikaz vrijednosti

Programska podrška

Praćenje i pohranjivanje podataka

Arduino pločica služi za spremanje vrijednosti senzora kao i izračun srednjih vrijednosti i standardne devijacije. Programska podrška za Arduino uključuje blago izmjenjenu DHT biblioteku koja je obavljala čitanje vrijednosti, prebacivanje i spremanje u varijable te provjeru kontrolne sume. Na početku se sve vrijednosti postavljaju na tisuću što omogućuje da se ta vrijednost nikada ne dostigne. Koristi se funkcija *resetData()*. Da je *default* vrijednost postavljena na nula, ne bi se moglo razaznati kada je nula izmjerena vrijednost, a kada *default*. U glavnom kodu svake dvije sekunde izvršava se prihvaćanje novih vrijednosti senzora. Vrijednosti kraće od dvije sekunde mogle bi remetiti očitanja, no u ovom slučaju češća očitanja nisu potrebna. Nakon očitanja vrijednosti, temperatura i vlažnost se spremaju u zasebne nizove. Korištena je logika kružnih lista i slanje trenutnog indeksa. Time se zna koja je vrijednost najnovija, a starije su vrijednosti redom lijevo od nje.

```
void setup() {
  Serial.begin(9600);
  resetData();
}

void loop() {
  delay(frequency);
  DHT.read11(dhtPin);

  addToMinuteArrays();
  addToHourArrays();

  printValues();

  checkCounters();
}
```

Slika 8. Glavni dio programa

```
void resetData(){
  lastMinuteCounter = 0;
  lastHourCounter = -1;
  addToHour = false;
  for(int i = 0; i < 30; i++){
    lastMinuteTemperatureArray[i] = 1000;
    lastMinuteHumidityArray[i] = 1000;
  }
  for(int i = 0; i < 60; i++){
    lastHourAverageTemperatureArray[i] = 1000;
    lastHourAverageHumidityArray[i] = 1000;
    lastHourSDTemperatureArray[i] = 1000;
    lastHourSDHumidityArray[i] = 1000;
  }
}
```

Slika 9. resetData()

```
void addToMinuteArrays(){
  lastMinuteTemperatureArray[lastMinuteCounter] = (int)DHT.temperature;
  lastMinuteHumidityArray[lastMinuteCounter] = (int)DHT.humidity;
}
```

Slika 10. Dodavanje novih vrijednosti

```
void addToHourArrays(){
  if(addToHour){
    addToAverageHours();
    addToSDHours();
  }
}
```

Slika 11. Poziv funkcija za dodavanje vrijednosti sata

```
float averageTemp(){
  float averageTemperature = 0;
  for(int i = 0; i < 30; i++)
    averageTemperature += lastMinuteTemperatureArray[i];
  return (averageTemperature / 30);
}
```

Slika 12. Računjanje srednje vrijednosti tempetarure

```
void addToAverageHours(){
  lastHourAverageTemperatureArray[lastHourCounter] = averageTemp();
  lastHourAverageHumidityArray[lastHourCounter] = averageHum();
}

void addToSDHours(){
  lastHourSDTemperatureArray[lastHourCounter] = SDTemp();
  lastHourSDHumidityArray[lastHourCounter] = SDHum();
}
```

Slika 13. Dodavanje vrijednosti sata

Nakon spremanja vrijednosti u nizove, isti se python kodu šalju preko serijske komunikacije. Kada Arduino zaprimi vrijednost "1" šalje vrijednosti za predhodnu minutu, a kada zaprimi "2" šalje za predhodnih sat vremena. U oba slučaja na kraju šalje i trenutni indeks polja na kojemu se nalazi.

```
void printValues(){
  if (Serial.available() > 0) {
    int incomingByte = Serial.read();

    if(incomingByte == 49){
      printMinute();
    }else if(incomingByte == 50){
      printHour();
    }
  }
}
```

Slika 14. Funkcija za ispis vrijednosti

```
void printMinute(){
  for(int i = 0; i < 30; i++){
    Serial.print(lastMinuteTemperatureArray[i]);
    Serial.print(",");
    Serial.print(lastMinuteHumidityArray[i]);
    Serial.print(";");
  }
  Serial.println(lastMinuteCounter);
}
```

Slika 15. Ispis vrijednosti

Grafičko sučelje

Python-ova programska podrška se temelji na bibliotekama *tkinter* (izrada prozora i objekata), *serial* (čitanje sa serijskog porta), *threading* (lakše upravljanje procesima i memorijom), *Matplotlib* (izračun i prikaz raznih grafova), *NumPy*, *pandas* i *re* (manipulacija podataka) te *time* i *datetime*(izračun i prikaz vremena).

Inicijalizacija serijskog porta kao i dretve koja obrađuje serijski ulaz i izlaz odrađuje se pritiskom na *Connect to Arduino* gumb. Za aktivaciju ulaza označujemo jedan od ponuđenih iz kombiniranog okvira. Istom logikom, pritiskom *Disonnect from Arduino* gumba zaustavlja se dretva, odnosno komunikacija.

```
def serialPorts():
    return [p.device for p in serial.tools.list_ports.comports()]
    pass

def serialInit():
    global serialPort
    serialPort = serial.Serial()
    serialPort.baudrate = 9600
    serialPort.port = COMPortCombobox.get()
    serialPort.open()

def startOrStop():...

def startSerialThread():
    global serialThreadBoolean
    if not serialThreadBoolean and not (COMPortCombobox.get() == 'None' or COMPortCombobox.get() == ''):
        serialThreadBoolean = True
        serialThread = Thread(target=serialInit, daemon=True)
        serialThread.start()
        return True
    else:
        error(0)
        return False

def stopSerialThread():
    global serialThreadBoolean
    global serialPort
    serialThreadBoolean = False
    serialPort.close()
```

Slika 16. Funkcije za čitanje i slanje sa serijskog porta, kao i za dretve

Opisani će biti samo proces koji se izvršava pritiskom na gumb *Show Last Minute* jer je za prikaz podataka sata jako sličan. Poziva se izravno funkcija *showMinuteStats()* koja šalje *"1"* na serijski port označavajući da želi primiti podatke o predhodnoj minuti. U slučaju da serijski port nije otvoren, dolazi do iznimke i program skočnim prozorom javlja da komunikacija između Arduina i računala nije započeta. Kada dobije podatke, odvaja vrijednosti u listu delimiterima *","* i *";"*. Iz te liste odvaja podatke temperature i vlage u zasebne nizove.

```
def showMinuteStats():
    global currentSecondsIndex
    try:
        if serialPort:
            serialPort.write(bytes("1", 'utf-8'))
            line = serialPortToLine()
            twoSeconds = re.split(';|,', line)
            y = 0
            for x in range(0, 60, 2):
                twoSecondsTemperature[y] = int(twoSeconds[x])
                twoSecondsHumidity[y] = int(twoSeconds[x+1])
                y += 1
            currentSecondsIndex = int(twoSeconds[60])

            g = rearangeMinutes()
            minuteStats(g)
            graphMinutes()
    except:
        error(1)
```

Slika 17. Dohvaćanje podataka

Poziva se funkcija *rearangeMinutes()* čiji je posao pronaći ispravan redoslijed podataka. Ista je krucijalna jer inače se nebi mogao odrediti ispravan poredak podataka, svaki bi mogao biti najnoviji. To radi uz pomoć funkcije *circularArray(a, l, ind)* koja, koristivši se pomoćnim

poljem, reorganizira podatke. Nakon namještanja ispravnog polja, u drugo se dodaju vremena očitavanja podataka. Funkcija na kraju vraća indeks zadnjeg elementa. Zatim se zove funkcija *minuteStats(g)* koja stvara novi prozor i u tablicu stavlja podatke nizova. Na kraju se poziva funkcija za stvaranje grafa koji iste vrijednosti prikaže grafički linearnim grafom.

```
def rearangeMinutes():
    global newTwoSecondsTemperature
    global newTwoSecondsHumidity
    global secondsTimeArray
    secondsTimeArray = [1000 for x in range(30)]
    tmp = [1000 for x in range(30)]
    g = 0
    t = 0

    tmp = circularArray(twoSecondsTemperature[::-1], 30, 29-currentSecondsIndex)
    newTwoSecondsTemperature = tmp[0:30]
    tmp = circularArray(twoSecondsHumidity[::-1], 30, 29-currentSecondsIndex)
    newTwoSecondsHumidity = tmp[0:30]

    if int(format(datetime.now(), '%S')) % 2 == 1:
        t = 1
    while g < 30 and newTwoSecondsTemperature[g] != 1000:
        secondsTimeArray[g] = format(datetime.now() - timedelta(seconds=g*2 - t), '%H:%M:%S')
        g += 1

    if g == 30:
        g = 29

    if secondsTimeArray[g] == 1000:
        secondsTimeArray = secondsTimeArray[:g]
        newTwoSecondsTemperature = newTwoSecondsTemperature[:g]
        newTwoSecondsHumidity = newTwoSecondsHumidity[:g]
    else:
        secondsTimeArray = secondsTimeArray[:g+1]
        newTwoSecondsTemperature = newTwoSecondsTemperature[:g+1]
        newTwoSecondsHumidity = newTwoSecondsHumidity[:g+1]

    return g
```

Slika 18. Manipulacija poljima

```
def circularArray(a, l, ind):
    b = [None]*2*l
    out = [None]*2*l
    i = 0

    while i < l:
        b[i] = b[l + i] = a[i]
        i += 1

    i = ind
    j = 0

    while i < l + ind :
        out[j] = b[i]
        i += 1
        j += 1

    return out
```

Slika 19. Namještanje kružnog niza

```
def graphMinutes():
    figureSeconds = plt.figure("Last minute", figsize=(9, 7))
    axisSeconds = figureSeconds.add_subplot()
    axisSeconds.set_xlabel('Values')
    axisSeconds.set_ylabel('Time')
    axisSeconds.plot(newTwoSecondsTemperature, secondsTimeArray, label='Temperature')
    axisSeconds.plot(newTwoSecondsHumidity, secondsTimeArray, label='Humidity')
    figureSeconds.subplots_adjust(top=0.97, left=0.121, right=0.95, bottom=0.088)
    axisSeconds.legend(loc='best', shadow=True)
    plt.show()
```

Slika 20. Stvaranje grafa

```
def showMinuteStats():
def showHourStats():
def minuteStats(g):
def hourStats(g):
def circularArray(a, l, ind):
def rearangeMinutes():
def rearangeHours():
def graphMinutes():
def graphHours():
```

Slika 21. Sve funkcije

Sažetak

U radu se upoznajemo načinom na koji smo realizirali mjerenje vrijednosti temperature i vlažnosti, koristeći se Arduinom i DHT11 senzorom, te grafičkim sučeljem napisanim u Python jeziku za vizualno prikazivanje istih. Dotičemo se razvojnog okruženja, sklopovlja i programske podrške korištene pri izradi projekta te detaljnije objašnjavamo neke dijelove koda i funkcije potrebne za ispravan rad projekta.

Literatura

https://store.arduino.cc/arduino-micro

https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

https://www.arduino.cc/en/software

https://github.com/arduino/Arduino

https://docs.python.org/3/library/tkinter.html

https://matplotlib.org/stable/index.html

https://numpy.org

https://docs.python.org/3/library/time.html