Vaja 7: Avtomatizacija zajema meritev v sistemu Arduino

Matej Bažec

8. januar 2019

Povzetek

Na primeru razvojne platforme Arduino bomo spoznali primer avtomatičnega zajema meritev. Najprej se bomo podrobneje seznanili z okoljem Arduino, preko katerega bomo poskušali predstaviti delovanje mikrokontrolerjev in logiko, ki stoji za njimi. Seznanili se bomo tudi z nekaj senzorji in z njimi bomo poskušali napraviti nekaj preprostih sistemov.

1 Platforma Arduino

Arduino je razvojna platforma, ki močno poenostavi programiranje mikrokontrolerjev. Fizično je sestavljena iz mikrokontorlerja (MCU - microcontroller unit) temelječega na arhitekturi AVR (največkrat sta to ATmega 328p in ATmega 2560) in komunikacijskega modula, ki skrbi za prenos serijske UART komunikacije preko USB vodila med mikrokontrolerjem in računalnikom. Preko tega kanala poteka izmenjava sporočil in programiranje Arduina. Ni pa ta povezava nujna za delovanje platforme.

Poleg same fizične konstrukcije sodi k Arduinu tudi programski del t. i. bootloader. To je manjši program, ki se požene ob zagonu Arduina. Njegova edina naloga je, da prenese morebitni program na Arduino. Če tega ni, požene že vnešen program. Slednje se zgodi vedno, ko je Arduino izklopljen od računalnika.

K sistemu Arduino sodi tudi integrirano razvojno okolje. To pa je program, ki teče na računalniku. V njem lahko pišemo kodo, jo prevajamo in nalagamo v MCU. Poleg tega skrbi še za knjigovodstvo sistemskih in uporabniških knjižnic.

1.1 Delovanje mikrokontrolerja

Mikrokontroler deluje podobno kot vsak računalnik. Izvaja dan program, ki je vpisan v njegov programski del spomina. V računalniškem smislu je precej omejen (tipično ima 16MHz delovnega takta, 32MB programskega spomina, 2kB delovnega spomina). Njegova prednost pa so direkten in poln dostop do vseh svojih vhodov/izhodov in popolna kontrola nad izvajanjem programa. To nam mogoča direktno branje in pošiljanje električnih signalov, ki so lahko digitalni ali analogni. Program se izvaja sekvenčno z izjemami prekinitev (interrupt), nad katerimi pa imamo popoln nadzor. To nam omogoča odziv v realnem času, kar je pri meritvah pomembno.

Stanje na vhodu preberemo tako, da preberemo stanje ustreznega registra (npr. PINB). Na podoben način postavimo stanje na izhodu s pisanjem v ustrezni register (npr. PORTB). Večini priključkov (pinov) v MCU lahko programsko določimo, ali bodo vhodni ali izhodni. To storimo tako, da vpišemo dano vrednost v ustrezen register (npr. DDRB). Pini so grupirani v skupine po 8 (izjemoma tudi manj), tako da z enim registrom naslavljamo do 8 pinov.

Tako npr. izgleda program, ki določi, da bosta pina PB0 in PB1 vhodna, PB2 pa izhodni. Vrednost na slednjem bo 5V le takrat, ko bosta oba PB0 in PB1 na visokem nivoju, sicer pa bo 0. Besedilo za podpičjem so komentarji.

```
ldi r24 , 0x4       ;0x4 pomeni tretji bit (PB2) je 1, ostali 0
out 0x04 , r24      ;0x04 je naslov registra PORTB
sbis 0x03, 0        ;0x03 je naslov registra PINB. Začetek zanke.
rjmp .+8            ;skok na brisanje bita - ukaz cbi
sbis 0x03, 1        ;0x03 je naslov registra PINB
rjmp .+4            ;skok na brisanje bita - ukaz cbi
sbi 0x05, 2         ;0x05 je naslov registra PORTB
rjmp .-12           ;skok na začetek zanke
cbi 0x05, 2         ;0x05 je naslov registra PORTB
rjmp .-16           ;skok na začetek zanke
```

Program je v zbirniku (assembler) v (namenoma) precej surovi obliki. Po navadi si pomagamo z oznakami (label) pri skokih in preddefiniranimi konstantami (pišemo PINB namesto 0x03). Kljub temu, je pisanje programa v zbirniku precej neprijetno in se ga z izjemo kritičnih delov izogibamo.

Zato se raje poslužujemo višjih programskih jezikov, največkrat C. Isti program bi tako zgledal v C (z uporabo knjižnice avr-libc).

```
#include <avr/io.h>

int main () {
        DDRB=1<<PB2;
        while (1){
                if (PINB&(1<<PB0) && PINB&(1<<PB1)) PORTB|=1<<PB2;
                else PORTB&=~(1<<PB2);
        }
}
```

1.2 Programiranje MCU v okolju Arduino

Okolje Arduino nam programiranje še naprej poenostavi. Pri Arduinu so namreč pini razvrščeni po vrstnem redu. Zato nam ni treba vedeti v kateri skupini ležijo. Tako npr. pinom PB0, PB1 in PB2 ustrezajo zaporedne številke 8, 9 in 10. Isti program v Arduinu bi izgledal tako.

```
void setup() {
        pinMode(8,INPUT);
        pinMode(9,INPUT);
        pinMode(10,OUTPUT);
}

void loop() {
        if (digitalRead(8) && digitalRead(9)) digitalWrite(10, HIGH);
        else digitalWrite(10,LOW);
}
```

Program je napisan v C++. Tisti, ki poznate klasični C++, boste opazili, da manjkajo ustrezni headerji (namenoma se izogibam prevoda v smislu vzglavje ali kaj podobnega), funkcija main, konstante niso nikjer definirane ... To je zato, ker je main že vključen v Arduinovo sistemsko knjižnico, od koder kliče funkciji setup in loop. Tudi Arduinovi sistemski headerji so avtomatično vključeni, zato so nekatere konstante (npr. `INPUT`) že definirane. Seveda imajo te poenostavitve svojo ceno. Program zaradi uporabe knjižnic teče bistveno počasneje, kot bi z direktnim naslavljanjem.

Kot vidimo iz zgornjega primera, ima vsak Arduino program dve funkciji tipa `void` brez argumentov. To sta `setup` in `loop`. Kot že imeni nakazujeta, v funkciji `setup` definiramo osnovne nastavitve in inicializiramo morebitne podsisteme. Ta funkicija se izvede le enkrat in sicer pri zagonu MCU. Po drugi strani, funkcija `loop`

teče v nedogled. Ko so vsi ukazi izvedeni, jo sistem kliče ponovno. Obe funkciji morata biti definirani, sta pa lahko prazni.

2 Senzorski moduli

Senzorje lahko na MCU priklopimo direktno, ali pa komuniciramo z nekim vmesnikom, ki skrbi za pravilno odčitavanje senzorja. Seveda moramo v prvem primeru sami poskrbeti, da bo senzor deloval v pravilnem režimu, pa tudi merjeno količino si moramo sami preračunati iz dobljene napetosti. Prikazali bomo delovanje v obeh načinih na primeru nekaj izbranih senzorjev, ki so primerno zapakirani prav za direktno uporabo v okolju Arduino.

2.1 Merjenje svetlosti

Modul za merjenje svetlosti deluje s pomočjo svetlobno odvisnega upornika (kar je ponesrečen prevod za light dependent resistor - LDR). Ta ima lahko zelo visoko upornost v temi (do 1MΩ) do zelo nizke (nekaj ohmov) pri močni osvetljenosti. Če ga priključimo zaporedno z znanim upornikom lahko s pomočjo napetostnega delilnika preračunamo izmerjeno napetost v upornost LDR-ja. Zato moramo pomeriti napetost analogno preko ADC-ja (analogno digitalnega pretvornika). Zato moramo priklopiti signalni pin na modulu z enim od analognih pinov na Arduinu. Primer programa za ta senzor je v imeniku "fotorezistor".

Primer je zelo preprost način, kako beremo analogne signale. Signal fizično priklopimo na enega od analognih vhodov, ki jih prepoznamo tako, da imajo črko A v imenu (v našem primeru A5). Nato analogni signal preberemo s funkcijo `analogRead(A5)`. Analogno digitalni pretvornik (ADC) nam pretvori napetost med 0 in 5V na vhodu na sorazmerno celoštevilčno vrednost med 0 in 1023 (ADC je 10-bitni), pri čemer predstavlja prvi 0, drugi pa 5V. Če želimo to vrednost pretvoriti v realno število, ki predstavlja volte, moramo torej pomnožiti s 5 in deliti s 1023.

Na tem primeru se bomo seznanili tudi z objektom `Serial`. To je objekt, preko katerega poteka serijska komunikacija preko protokola UART. V Arduinu je predefiniran, zato ga ni treba posebej deklarirati. V naših primerih bomo uporabljali tri metode: `begin`, `print` in `println`. S prvim inicializiramo UART podsistem in nastavimo baudrate (tega se niti ne upam prevesti). Z drugima dvema pa izpisujemo besedilo, pri čemer `println` na koncu teksta preide v novo vrstico.

Funkcija `delay` ustavi itvajanje programa za določeno število milisekund.

2.2 Senzor bližine

Modul za merjenje bližine deluje na principu odboja ultrazvočnega valovanja. Če želimo izmeriti razdaljo, postavimo pin Trig na 5V za kratek čas (npr. 10μs). Ko postavimo Trig nazaj na 0V, bo modul postavil pin Echo na 5V in oddal kratek ultrazvočni signal frekvence 40kHz in dolžine 8 period. Ta se bo odbil od ovire in pripotoval na mikrofon na modulu. Ko bo modul zaznal odbit signal, bo vrnil pin Echo na 0V. Če poznamo hitrost zvoka (343m/s), lahko iz časa potovanja preračunamo razdaljo.

Primer ne uporablja nobene knjižnice, zato moramo vse implementirati sami. Pin, ki ga bomo povezali s Trig pinom na modulu, bomo definirali kot izhodni (`OUTPUT`). Drugi pin, ki ga bomo povezali z Echo, pa kot vhodni (`INPUT`). Za ta namen bomo uporabili funkcijo `pinMode`. Na prožilni pin bomo pisali s funkcijo `digitalWrite`. Funkcijo `delayMicroseconds` bomo uporabili za čakanje 10μs. Dolžino trajanja pa bomo izmerili s funkcijo `pulseIn`.

2.3 Temperaturni senzor

Modul za merjenje temperature uporablja Dallasov DS18B20 senzor. Ta senzor že sam pretvori temperaturo v digitalno obliko in jo odda preko onewire komunikacijskega protokola. Onewire je dokaj pogost protokol, ni pa tako popularen, da bi bil že vnaprej vključen v Arduinovo knjižnico. Zato moramo dodati v imenik `libraries` knjižnico `onewire`, ki nam definira header `OneWire.h`, v katerem je definiran razred `OneWire`. MCU, ki jih uporablja Arduino, nimajo vgrajenega onewire protokola hardversko (za razliko od npr. UART-a), zato je treba protokol implementirati z direktnim branjem ozirom pisanjem na ustrezni pin. Takemu pristopu pravimo v žargonu bitbanging.

Da si olajšamo branje temperature s senzorja uporabimo še knjižnico `dallas`, ki implicitno uporablja še knjižnico `adafruit`. Ta nam definira razred `DallasTemperature`. Objekti tega tipa direktno berejo s senzorja temperaturo preko metod `requestTemperatures` in `getTempCByIndex`. Prva prebere temperaturo in jo shrani, druga pa pretvori shranjeno vrednost v stopinje celzija.

2.4 Senzor temperature in vlage

Ta modul vsebuje senzor DHT11. Ta senzor je spet digitalni, tako da beremo direktno vrednosti v digitalni obliki. Senzor uporablja svoj lasten protokol (ki je sicer precej podoben onewire), za katerega pa ne rabimo skrbeti, saj lahko uporabimo knjižnico `dht`. Ta nam definira header `DHT.h`, preko njega definiramo objekt razreda DHT. Objekt inicializiramo z metodo `begin`, temperaturo beremo z `readTemperature` (v stopinjah celzija), vlago pa z `readHumidity` (v odstotkih relativne vlažnosti). V primeru napake, funkciji vrneta `Nan` (not a number).

2.5 Senzor temperature in tlaka

Ta modul uporablja zelo natančen senzor BMP280, ki je digitalni. komunicira lahko preko dveh standardnih protokolov: SPI in I²C (Atmel/Microchip uporablja oznako TWI zaradi licenčnih razlogov). Knjižnica `bmp280` omogoča komunikacijo s senzorjem na tri načine, ki se jih določi ob definiciji objekta razreda `Adafruit_BMP280`: hardverski I²C (konstruktor kličemo brez argumentov), hardverski SPI (en argument - CS pin) in softverski SPI (štirje argumenti).

Surov signal je precej neobdelan in je potreben precejšnje računske obdelave, vključno z upoštevanjem kalibracijskih parametrov. Na srečo je preračunavanje skrito v knjižnici. Tako lahko do temperature in tlaka dostopamo s preprostim klicem metod `readTemperature` (v stopinjah celzija) in `readPressure` (v Pa).

2.6 Mikrofon

Modul za merjenje zvoka je sestavljen iz mikrofona, ojačevalca in diskriminatorja. Lahko beremo direktno analogno vrednost na mikrofonu, ali pa beremo digitalno vrednost na diskriminatorju. Ta bo dal 0, če bo nivo zvoka pod dano vrednostjo, in 1, če bo nad njo. Vrednost lahko mehansko nastavimo z vijakom na potenciometru.

V funkciji `setup` najprej definiramo oba pina (analognega in digitalnega) kot vhoda s funkcijo `pinMode` (to sicer ni nujno potrebno, ker je privzeta vrednost na pinih `INPUT`). `analogRead` smo že srečali pri modulu za merjenje svetlosti, `digitalRead` pa lahko vrne le vrednosti 0 oziroma 1.

3 Naloge

Na vajah vam bo asistent določil, katero od omenjenih vaj naredite doma. Nalogo prinesite na naslednjih vajah, kjer jo boste predstavili sošolcem.

3.1 Naloga 1

S pomočjo modulov za merjenje bližine in aktivnega zvočnika naredite napravo, ki bo zapiskala vsakič, ko bo zaznala objekt na razdalji manjši od 25cm.

3.2 Naloga 2

S pomočjo modulov za merjenje temperature in vlage in releja naredite napravo ter ventilatorja naredite napravo, ki bovključila ventilator vsakič, ko bo temperatura presegla 26°C, ali ko bo relativna vlažnost presegla 80%.

3.3 Naloga 3

Povežite modula za merjenje temperature in dvobarvno diodo tako, da bo svetila pod 22°C zeleno, do 25°C rumeno, do 28°C oranžno in nad tem rdeče.

3.4 Naloga 4

Isto kot naloga 3, le da uporabite senzor za tlak in temperaturo za merjenj temperature.

3.5 Naloga 5

Napravite model avtomatične žarnice. Ko bo na svetlobnem senzorju majhna osvetlitev, prižgite diodo.

3.6 Naloga 6

S pomočjo modula za merjenje zvoka in modula KY-027 napravite napravo, ki bo zasvetila vsakič, ko bo dan nivo hrupa presežen.

3.7 Dodatne naloge

Če želite, lahko uporabite kakšen drugi modul iz istega kompleta, ali predlagate svoj načrt.

Literatura

[1] Referenčna dokumentacija za Arduino. `https://www.arduino.cc/reference/en/`.

[2] TkkrLabova spletna dokumentacija. `https://tkkrlab.nl/wiki/Arduino_37_sensors`.

[3] Uporabljeni primeri in knjižnice. `http://vaje.fpp.uni-lj.si/tm/arduino.tar.bz2`.