9A0BVP – rkp.beacon

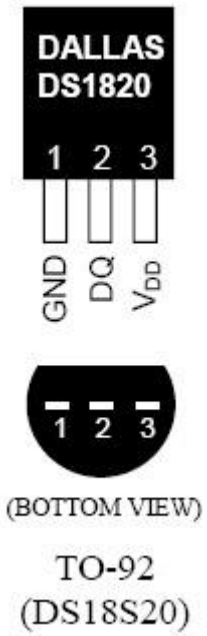
Beacon

Beacon, automatski odašiljač, u radioamaterskom svijetu koristi se za analizu propagacija. On neprekidno odašilje svoju pozivnu oznaku i lokaciju, kako bi radioamateri koji ga primaju mogli znati što slušaju. Na osnovu te informacije, znaju s kojim radioamaterima mogu u tom trenutku najlakše uspostaviti vezu. Beacon danas ima mnoštvo, i postavljeni su na raznim lokacijama s antenama usmjerenima u raznim smjerovima, kako bi analize propagacija bile što jednostavnije.

Osnovni dijelovi beacona najčešće su odašiljač (često snage manje od 1W) i antena (najčešće dipol) i mikrokontroler. U ovom će tekstu biti opisan mikrokontrolerski dio 9A0BVP beacona.

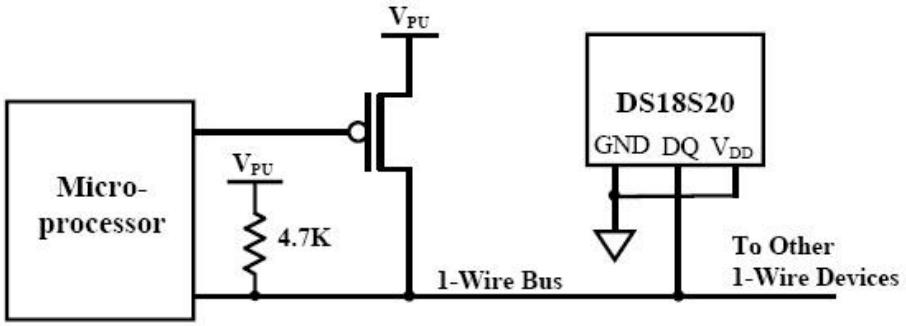
Digitalna temperaturna sonda

Digitalna se tehnologija svakim danom sve više razvija, što se odražava i na cijenu uređaja i komponenata i njihovu popularnost i dobavljivost. Senzori sa razne veličine dostupni su kao digitalne komponente koje unutar sebe stvarnu veličinu pretvore u električnu, koju tada digitaliziraju. «Kap u moru» je mjerenje temperature. Digitalna temperaturna sonda **Dallas DS18S20** može se za pedesetak kuna kupiti u skoro svakoj trgovini elektroničkog materijala. Njezine karakteristike su točnost od ±0.5°C, temperaturni raspon od -55°C do 125°C, napon napajanja od 3V do 5V, vrijeme pretvaranja 750 ms. Navedena sonda podržava 1-Wire rad (potrebne su joj samo dvije žice: masa i napajanje/podaci –može se napajati iz linije za prijenos podataka), ali ga u ovom projektu nećemo koristiti.



Prvi korak u korištenju ovakve sonde je čitanje (s razumijevanjem) njenog uputstva (*datasheet*, koji je dostupan na stranicama proizvođača: www.maxim-ic.com).

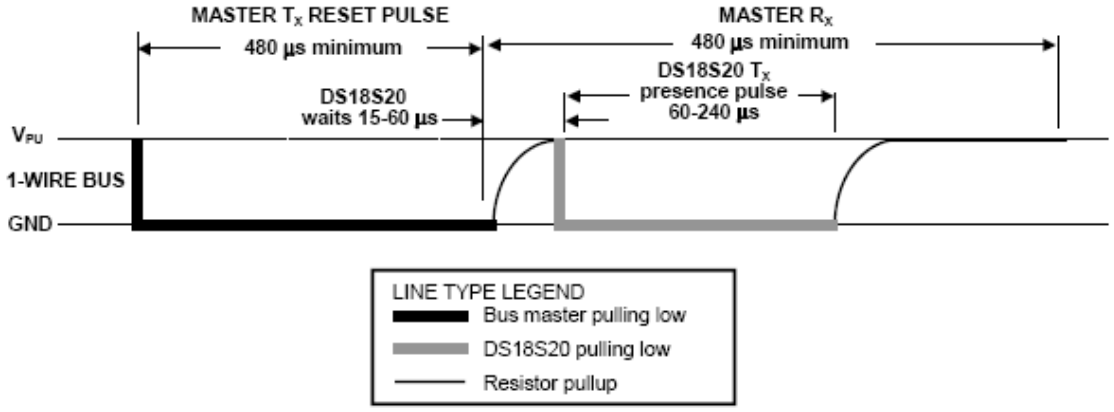
Elektronički dio je vrlo jednostavan. Sondu treba spojiti na napon napajanja (5V) i masu, i podatkovnu nožicu s PICem. Na podatkovnu (u ovom slučaju srednju) nožicu treba spojiti *pull-up* otpornik prema naponu napajanja. Taj će otpornik osigurati ispravni rad sonde. Kada sonda piše podatke u svoje EEPROM ili kada obavlja pretvorbu temperature, struja može narasti na 1.5 mA. Takva struja može prouzročiti preveliki pad napona na internom *pull-up* otporniku, te sonda nebi imala dovoljno energije za nastavak rada. Ovo se posebno odnosi na sondu u 1-Wire načinu rada, a otpornik je poželjno staviti neovisno o načinu napajanja sonde. Napon V_{PU} je napon napajanja sonde, tj. 5V.



Digitalna komunikacija prema sondi je jednostavna: na u uputstvima opisani način se sondi pošalje naredba (veličine 1 byte), na koji sonda odgovara podacima. Vrlo je važno izraditi ispravne rutine za komunikaciju prema sondi (za slanje i primanje bitova). Sonda ima točno definirana vremena trajanja pojedinog stanja na njenoj podatkovnoj (DQ) nožici i toga se treba strogo držati.

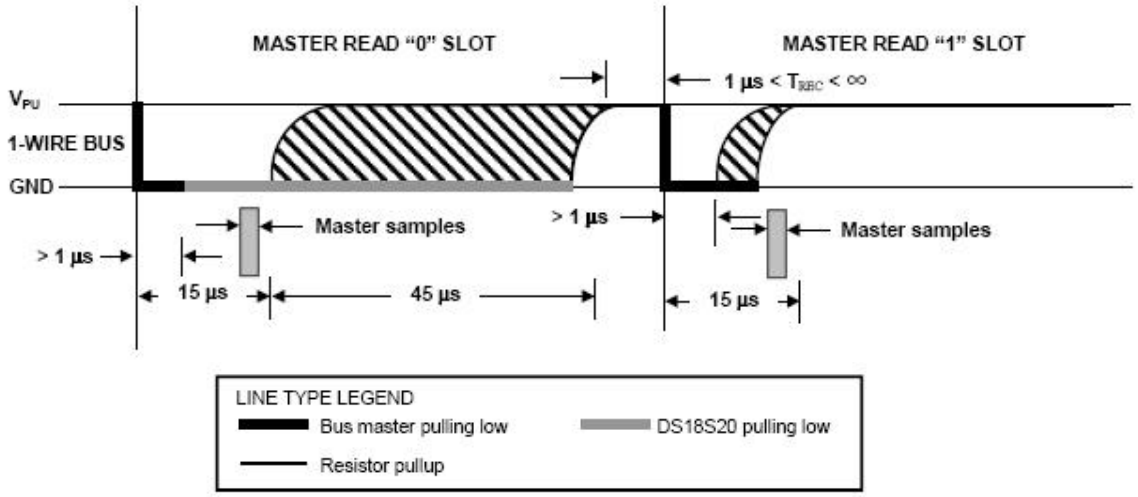
Inicijalizacija sonde (tj. inicijalizacija komunikacije s njom) se vrši tako da se pin na koji je sonda spojena postavi na logičku 0 najmanje 480 µs (*reset pulse*). Zatim se počeka barem 60 µs i pročita stanje tog pina. Ukoliko je sonda spremna za rad i sve je u redu, ona odgovara postavljanjem pina na logičku jedinicu (*presence pulse*). Sondi možemo slati naredbe ili čitati podatke s nje tek kad je inicijalizacija uspješno obavljena.

INITIALIZATION TIMING Figure 10



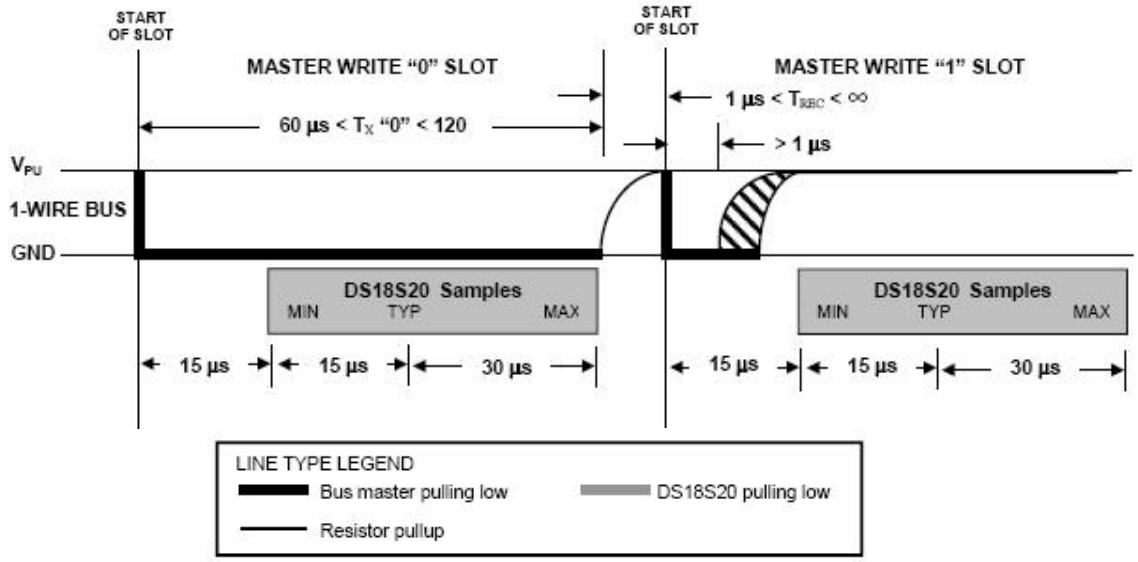
Kada želimo poslati naredbu sondi, mi joj zapravo pošaljemo 8 bita. Slanje bitova se izvodi na slijedeći način. Pin na koji je sonda spojena postavimo na logičku 0 barem 1 µs, zatim pin postavimo na logičku 0 ili logičku 1, ovisno o bitu koji šaljemo, u trajanju od barem 60 µs, i na kraju postavimo pin na logičku jedinicu barem 1 µs.

WRITE TIME SLOT TIMING DIAGRAM



Čitanje se izvodi na sličan način. Pin na koji je sonda spojena postavimo na logičku 0 barem 1 µs. Od toga trenutka imamo najviše 15 µs da pročitamo bit koji nam sonda daje (postavlja). Nakon tih 15 µs sonda postavlja vrijednost logičke 1. Na kraju počekamo 60 µs da se završi ciklus slanja bita.

READ TIME SLOT TIMING DIAGRAM

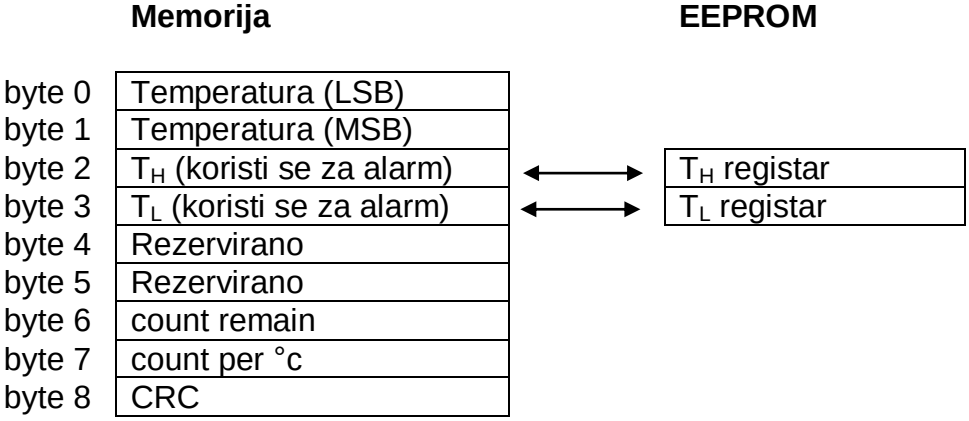


Ove je rutine najbolje spremiti u posebnu datoteku, kako bi sam kod bio što pregledniji i čitljiviji. Uspješnim dovršavanjem ovih rutina riješili smo veći dio problema komunikacije sa sondom. Slijedi zanimljiviji dio: slanje naredbi za pretvorbu temperature i njeno čitanje.

Naredba	**Kod**	**Značenje / objašnjenje**
Convert T	44h	Početak temperaturne konverzije
Read Scratchpad	BEh	Čitanje cijele memorije, uključujući CRC byte
Write Scratchpad	4Eh	Zapisuje podatke u memoriju (byte-ovi 2 i 3, tj. T_H i T_L)

Copy Scratchpad	48h	Pročita vrijednosti T_H i T_L iz memorije i zapisuje ih u EEPROM (koristi se za alarm)
Recall E^2	B8h	Pročita vrijednosti T_H i T_L iz EEPROMa i zapisuje ih u memoriju (koristi se za alarm)
Read Power Supply	B4h	Pročita način napajanja (V_{CC} ili 1-Wire)

Izgled memorije prikazan je slijedećom tablicom:



Sonda u svoja prva dva bajta (byte 0 i byte 1) daje 9 bita podataka. Ti podaci izgledaju kao u slijedećoj tablici:

TEMPERATURA	DIGITAL OUTPUT (Bin)	DIGITAL OUTPUT (Hex)
+85.0°C	0000 0000 1010 1010	00AAh
+25.0°C	0000 0000 0011 0010	0032h
+0.5°C	0000 0000 0000 0001	0001h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1111	FFFFh
-25.0°C	1111 1111 1100 1110	FFCEh
-55.0°C	1111 1111 1001 0010	FF92h

Negativna temperatura zapisana je koristeći drugi komplement. Da bi se iz digitalnih podataka dobila temperatura, treba dobiveni broj podijeliti s vrijednosti najmanje značajnog bita (LSB). To je u našem slučaju (9 bitna konverzija) 2. Npr. digitalni zapis 0000 0000 0000 0010 je u dekadskom sustavu broj 2. Broj 2 podijelimo s vrijednosti LSB-a, tj. s 2, i dobijemo 1. Ovo znači da digitalni zapis 0000 0000 0000 0010 prikazuje temperaturu od 1°C.

Vrijednosti *count remain* i *count per °c* služe za dobivanje podatka o vrijednosti temperature u višoj (12-bitnoj) rezoluciji. Tada nije dovoljno pročitati samo ta dva byte-a, već treba koristiti sljedeću formulu:

$$temperatura = temp\_read - 0.25 + \frac{count\_per\_c - count\_remain}{count\_per\_c}$$

count_per_c i *count_remain* su vrijednosti sedmog i šestog byte-a (byte7 i byte 6), a temp_read je vrijednost temperature dobivena čitanjem prva dva byte-a (byte 0 i byte 1). Koristeći ovu metodu može se dobiti u 12-bitnoj rezoluciji, što znači da je najmanja detektirana promjena temperature 0.03125°C. Koristeći ovu rezoluciju dobiti ćemo mala «osciliranja» vrijednosti, jer sonda nije originalno zamišljena da radi na toj rezoluciji.

Program za PIC pisati ćemo u C-u, a za prevoditelj ćemo koristiti CCS C compiler, čija se besplatna inačica može skinuti sa njihovih web stranica (www.ccsinfo.com).

Krenimo sa postavljanjem definicija. Da ne moramo svaki put u kodu pisati broj pina na koji je spojena DS18S20 sonda, nazvati ćemo ga ONE_WIRE_PIN:

```
#DEFINE ONE_WIRE_PIN PIN_A0 // sonda je spojena na PIN_A0
```

Zatim ćemo definirati imena registara unutar DS18S20 sonde. To će nam olakšati kasniji rad, jer je lakše zapamtiti funkcije pojedinog memorijskog byte-a, nego njegov redni broj u memoriji.

```
// ds1820 scratchpad registri
#define DS1820_SP_TLSB  0       // najmanje značajan bit temperature
#define DS1820_SP_TMSB  1       // najznačajniji bit temperature
#define DS1820_SP_HLIM  2       // HLIM (za alarm)
#define DS1820_SP_LLIM  3       // LLIM (za alarm)
#define DS1820_SP_CFG   4       // rezervirano
#define DS1820_SP_RES0  5       // rezervirano
#define DS1820_SP_CR    6       // count_remain
#define DS1820_SP_CPC   7       // count_per_c
#define DS1820_SP_CRC   8       // CRC kontrola greške
```

Po istom ćemo principu definirati i naredbe sonde.

```
// ds1820 command set
#define DS1820_CMD_READROM          0x33
#define DS1820_CMD_SKIPROM          0xCC
#define DS1820_CMD_CONVERTTEMP      0x44
#define DS1820_CMD_WRITESCRATCHPAD  0x4E
#define DS1820_CMD_READSCRATCHPAD   0xBE
#define DS1820_CMD_COPYSCRATCHPAD   0x48
```

Objašnjenja svake naredbe dana su već u ovom članku, pa ih nećemo ovdje ponavljati.

Ono što smo već prije spomenuli kao vrlo važnu stvar, praćenje točno zadanih vremena za inicijalizaciju sonde, kao i za slanje podataka sondi i čitanje rezultata sa nje, ovdje ćemo prikazati dijelovima koda, u obliku funkcija

Važno je spomenuti da pin na koji je spojena sonda mora biti i ULAZ u PIC i IZLAZ iz PICa. Njegova će se stanja mijenjati u kodu, i to će biti napomenuto.

Inicijalizaciju sonde je najbolje napraviti tako da se odmah provjeri jeli ona uspješno prošla, ili je pri inicijalizaciji nastala greška (što znači da ne možemo pristupi sondi).

```
short int onewire_init_with_error_check()
{
   onewire_disable_interrupts(TRUE);      // onemogućimo prekide
   output_low(ONE_WIRE_PIN);         // postavimo pin na logičku nulu
   delay_us( 500 );                  // 500 mikrosekundi
   output_float(ONE_WIRE_PIN);       // bit postaje ULAZ u PIC
   delay_us( 5 );                    // 5 mikrosekundi za stabilizaciju
   if (!input(ONE_WIRE_PIN))
      {
      onewire_disable_interrupts(FALSE); // omogućimo prekide
      return ( FALSE );       // greška (npr. sonda kratkospojena)
      }
   delay_us( 60 );      // pričekamo puls prisutnosti
   if (input(ONE_WIRE_PIN)) {
      onewire_disable_interrupts(FALSE); // omogućimo prekide
      return ( FALSE ); // sonda nije priključena
      }
   delay_us( 500 ); // pričekamo kraj ciklusa inicijalizacije
   output_float(ONE_WIRE_PIN);       // pin postaje ULAZ u PIC
   onewire_disable_interrupts(FALSE);     // omogućimo prekide
   return ( TRUE ); // sonda je spojena i spremna za rad
}
```

Iz koda se primijećuje da su neka vremena malo izmijenjena. Razlog tome je ispitivanje rada sonde i isprobavanje sa tim vrijednostima, koje su se pokazale kao stabilne (s tim se vrijednostima, uz neprekidno mjerenje, dobivaju točni podaci).

Za slanje podataka sondi koristimo sljedeću funkciju:

```
void onewire_sendbyte(int data)
{
   int count;
         onewire_disable_interrupts(TRUE); // onemogućimo prekide
   for (count=0; count<8; ++count)
      {
      output_low(ONE_WIRE_PIN);     // postavimo bit na logičku nulu
      delay_us( 2 );                // 2 mikrosekunde
      output_bit(ONE_WIRE_PIN, shift_right(&data,1,0)); // postavimo bit
      delay_us( 80 ); // pričekamo kraj ciklusa zapisivanja
      output_float(ONE_WIRE_PIN); // postavimo bit na logičku jedinicu
      delay_us( 2 );          // 2 mikrosekunde, što označava kraj ciklusa
      } // kraj for petlje
   onewire_disable_interrupts(FALSE);     // opet omogućimo prekide
} // kraj funkcije
```

Kod zapravo samo prati definirane vremenske intervale i unutar njih postavlja bit na logičku 0 ili 1. preko varijable *data* zadajemo byte koji želimo poslati sondi. *For* petlja se izvrši 8 puta, jer jedan byte ima 8 bita.

Analogno funkciji za slanje, definiramo i funkciju za čitanje podataka sa sonde.

```
int onewire_readbyte()
{
   int count, data;
         onewire_disable_interrupts(TRUE); // onemogućimo prekide
   for (count=0; count<8; ++count)
      {
      output_low(ONE_WIRE_PIN);     // postavimo pin na logičku 0
      delay_us( 2 );                // 2 mikrosekunde
      output_float(ONE_WIRE_PIN);   // pin postaje ULAZ u PIC
      delay_us( 10 );               // 10 mikrosekundi za stabilizaciju
      shift_right(&data,1,input(ONE_WIRE_PIN)); // pročitamo vrijednost
      delay_us( 60 );               // počekamo kraj ciklusa čitanja
      } // kraj for petlje
      onewire_disable_interrupts(FALSE); // ponovno omogućimo prekide
   return( data );
} // kraj funkcije
```

Četvrta važna funkcija je funkcija kojom ćemo pročitati rezultate pretvorbe temperature. Temperaturna sonda DS18S20 ne šalje samo podatke koje mi želimo, nego cijeli njezin *scratchpad*, koji se sastoji od 9 byte-ova koje smo prije opisali. *Scratchpad* je sondina interna memorija koja se sastoji od 9 registara veličine po 1 byte svaki. Za čitanje cijelog *scratchpad*a pozvati ćemo 9 puta funkciju za čitanje 1 byte-a sa sonde, odmah nakon što izdamo naredbu za čitanje.

```
void onewire_ds1820_read_scratchpad(int *data)
{
   onewire_init_with_error_check();// inicijaliziramo sondu
   onewire_sendbyte(DS1820_CMD_SKIPROM); // preskočimo ROM memoriju
   onewire_sendbyte(DS1820_CMD_READSCRATCHPAD);// pročitamo scratchpad

   delay_ms(2);   // počekamo 2 milisekunde zbog stabilizacije

   data[DS1820_SP_TLSB]=onewire_readbyte();    // 0 TLSB
   data[DS1820_SP_TMSB]=onewire_readbyte();    // 1 TMSS
   data[DS1820_SP_HLIM]=onewire_readbyte();    // 2 HLIM
   data[DS1820_SP_LLIM]=onewire_readbyte();    // 3 LLIM
   data[DS1820_SP_CFG]=onewire_readbyte();     // 4 reserved
   data[DS1820_SP_RES0]=onewire_readbyte();    // 5 reserved
   data[DS1820_SP_CR]=onewire_readbyte();      // 6 COUNT_REMAIN
   data[DS1820_SP_CPC]=onewire_readbyte();     // 7 COUNT_PER_C
   data[DS1820_SP_CRC]=onewire_readbyte();     // 8 CRC
}
```

Uz pročitane vrijednosti nadopisani su i komentari, da se lakše razumije koji se byte prenosi, iako je to vrlo intuitivno iz samog imena varijabli.

U samu funkciju sa varijabla *data* prenosi kao pokazivač na memorijski prostor (za potpuno razumijevanje najbolje je pogledati kako pokazivači rade u programskom jeziku C). Time smo uštedjeli na memoriji, i sama varijabla je vidljiva u funkciji i u glavnom programu. Ovdje posebno dolaze do izražaja korisnosti definiranja «zvučnih» imena varijabli: podaci se referenciraju kao njihova imena, a ne kao broj byte-a.

Ovim smo funkcijama potpuno pokrili rad sa sondom. Poželjno ih je izdvojiti iz koda i spremiti u zasebnu datoteku. Tako će ostatak koda biti pregledniji i čitljiviji.

Sada dolazimo do dijela gdje objedinjujemo sve zajedno: inicijalizacija → slanje naredbe sondi da u svoje registre pohrani podatak o temperaturi (pretvorba temperature) → čitanje registara → punjenje varijabli -→ kucanje telegrafijom.

```
float onewire_ds1820_read_temp_c_extended ()
{
// definiramo varijable i tipove varijabli koje ćemo koristiti
   int temperatureLSB, temperatureMSB, scratchpad[9];
   int count_remain, count_per_c;
// temperatura je 9 bitna, i može biti pozitivna ili negativna
// vrijednost int je 8 bitna (int8), pa će nam trebati 16-bitni int
   signed int16 temp;
// ako ne možemo inicijalizirati sondu, prekinemo izvođenje
   if (!onewire_init_with_error_check())     return (0);
// očitamo temperaturu i podatke spremimo u memoriju, preskačemo
// čitanje ROM memorije
   onewire_sendbyte(DS1820_CMD_SKIPROM);
   onewire_sendbyte(DS1820_CMD_CONVERTTEMP);
// počekamo 1 sekundu da se izvrši pretvorba
      delay_ms(1000);
// pročitamo registre (memoriju)
   onewire_ds1820_read_scratchpad(scratchpad);
// pretvorimo dobivene byte-ove u broj koji pokazuje temperaturu
   temperatureLSB = scratchpad[DS1820_SP_TLSB];
   temperatureMSB = scratchpad[DS1820_SP_TMSB];
      temp=(signed int16)make16(temperatureMSB, temperatureLSB);
// tražimo i varijable za 12-bitni oblik temperature
   count_remain   = scratchpad[DS1820_SP_CR];
   count_per_c    = scratchpad[DS1820_SP_CPC];
// Izračunamo temperaturu koristeći formulu iz uputstava sonde
precise = (float)temp*0.5 - 0.25 + (count_per_c - count_remain) /
(float)count_per_c;
// kraj funkcije
return (precise);
}
```

Zadnja funkcija u nizu funkcija za rad sa digitalnom sondom je funkcija koja će za vrijeme komunikacije sa sondom onemogućiti prekide (signale sa ostalih i na ostale portove). Ovo je važno, da bismo dobili komunikaciju sa sondom bez greške.

```
void onewire_disable_interrupts(int disable) {
   if (disable)
     disable_interrupts(GLOBAL);
   else
     enable_interrupts(GLOBAL);
}
```

Nakon što smo definirali funkcije za rad s digitalnom sondom, moramo definirati funkcije za kucanje telegrafije. Kako znamo da je telegrafija kombinacija dugih i kratkih tonova, pri čemu jedan dugi ton traje koliko jedan kratak, a pauza

između slova traje kao jedan dugi ton, napisati ćemo tri odvojene funkcije (točka, crta i pauza).

```
void pause(int koliko)
{
   int a;
   for (a=0;a<koliko;a++) { delay_ms(delay); }
}


void dit(void)
{
   output_high(CW_KEYER);
      pause(1);
   output_low(CW_KEYER);
      pause(1);
}


void dah(void)
{
   output_high(CW_KEYER);
      pause(3);
   output_low(CW_KEYER);
      pause(1);
}


void key_down(void)
{
   output_high(CW_KEYER);
      delay_ms(20000);
   output_low(CW_KEYER);
      pause(3); pause(3);
}
```

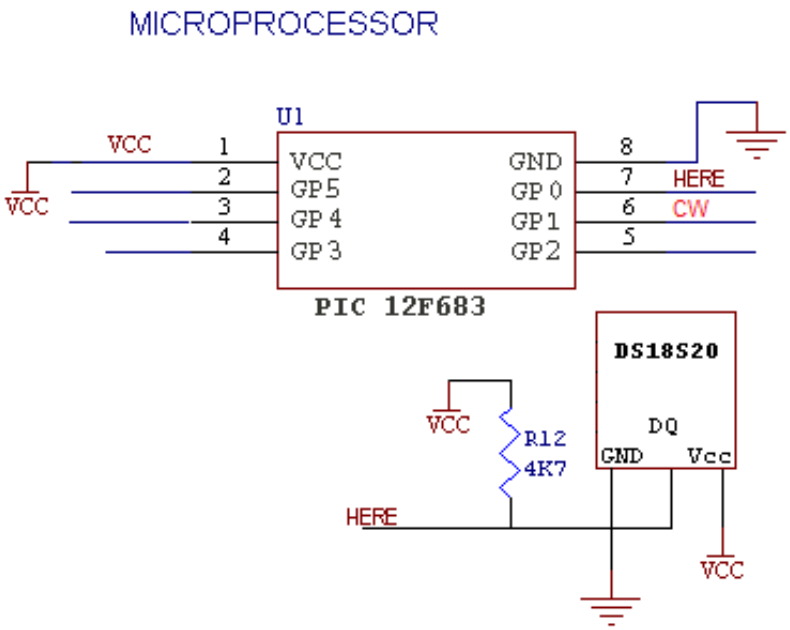
Slijedi definiranje pojedinog znaka (slova, brojke ili interpunkcije). Npr. slovo C se sastoji od točke, crte, točke, crte (i to baš tim redoslijedom). To znači da funkcije za točku i crtu treba pozvati tim redom, i nakon toga pozvati funkciju za pauzu.

```
void cw_send(char znak)
{
   switch (znak)
   {
      case '0': dah(); dah(); dah(); dah(); dah(); pause(3); break;
      case '1': dit(); dah(); dah(); dah(); dah(); pause(3); break;
      case '2': dit(); dit(); dah(); dah(); dah(); pause(3); break;
      case '3': dit(); dit(); dit(); dah(); dah(); pause(3); break;
      case '4': dit(); dit(); dit(); dit(); dah(); pause(3); break;
      case '5': dit(); dit(); dit(); dit(); dit(); pause(3); break;
      case '6': dah(); dit(); dit(); dit(); dit(); pause(3); break;
      case '7': dah(); dah(); dit(); dit(); dit(); pause(3); break;
      case '8': dah(); dah(); dah(); dit(); dit(); pause(3); break;
      case '9': dah(); dah(); dah(); dah(); dit(); pause(3); break;

      case '.': dit(); dah(); dit(); dah(); dit(); dah(); pause(3);
break;
      case '-': dah(); dah(); dit(); dit(); dah(); dah(); pause(3);
break;
      case '/': dah(); dit(); dit(); dah(); dit(); pause(3); break;

      case ' ': pause(3); break;

      case 'p': pause(3); break;

      case 'A': dit(); dah(); pause(3); break;
      case 'B': dah(); dit(); dit(); dit(); pause(3); break;
      case 'C': dah(); dit(); dah(); dit(); pause(3); break;
      case 'E': dit(); pause(3); break;
      case 'F': dit(); dit(); dah(); dit(); pause(3); break;
      case 'H': dit(); dit(); dit(); dit(); pause(3); break;
      case 'J': dit(); dah(); dah(); dah(); pause(3); break;
      case 'N': dah(); dit(); pause(3); break;
      case 'K': dah(); dit(); dah(); pause(3); break;
      case 'P': dit(); dah(); dah(); dit(); pause(3); break;
      case 'R': dit(); dah(); dit(); pause(3); break;
      case 'V': dit(); dit(); dit(); dah(); pause(3); break;
      case 'W': dit(); dah(); dah(); pause(3); break;
      case 'X': dah(); dit(); dit(); dah(); pause(3); break;
   } // end switch
} // kraj funkcije
```

Iz funkcije se vidi da nisu definirani svi znakovi, nego samo oni koji nam trebaju za kucanje temperature i uvodne poruke. Zbog ograničenja memorije u korištenom PICu, upisani su samo znakovi koji se koriste.

Sada kada potpuno znamo i razumijemo kako «stvar» radi, možemo pristupiti praktičnoj izvedbi koja je vrlo jednostavna. Shema je prikazana sljedećom slikom.

Praktična izvedba



Podaci o PICu **12F683**:
Flash memorija, 3.5 KB programske memorije, 2048 riječi, 256 B podatkovne memorije, 128 B RAM, 4-kanalni 10 bitni AD konvertor, 1 komparator, 8bitni i 16bitni timeri, maksimalna frekvencija: 20 MHz, 8 pinova.

Spajanje sklopa vrlo je jednostavno: prema već objašnjenim uputama sonda se spaja s mikrokontrolerom, a njemu je potrebno samo napajanje. Na nožici označenoj s „CW“ dobivamo napajanje koje zatim koristimo kao ulaz u optocopupler ili tranzistor za preklapanje predaje odašiljača, no o tome će više biti riječi u narednom dokumentu.

U nastavku je dan kod programa. Funkcije koje se koriste u programu već su objašnjene u ovom tekstu, pa o njima neće biti posebno riječi kod opisa programa. Ovdje su prikazane definicije varijabli, funkcija za onemogućavanje prekida (poželjno je onemogućiti prekide rada mikrokontrolera za vrijeme komuniciranja sa sondom) i glavna funkcija. Gore opisane funkcije spremljene su u datoteku ***ds18s20.c*** koju u glavnom programu moramo uključiti.

Jednom kad smo funkcije za rad sa sondom spremili u odvojenu datoteku, rad i debuggiranje su znatno olakšani.

Brzina kucanja telegrafije definirana je varijablom *delay*. To je vrijeme trajanja jedne 'točke', izraženo u milisekundama. Formula za pretvaranje iz 'znakova u minuti (*cpm*)' ili 'riječi u minuti (*wpm*)' u varijablu *delay* piše u funkciji, a glasi:

$$delay = \frac{60 \cdot 5 \cdot 100}{46 \cdot cpm} \qquad delay = \frac{60 \cdot 100}{46 \cdot wpm}$$

Drugi način računanja ovo formule je sljedeći: za otkucati PARIS (standard za izračunavanje „riječi u minuti“ potrebno je 165 perioda (jedan period je jedna točka, a uključene su i praznine)). Ako jednu minutu (6000 milisekundi) podijelimo s 165, dobijemo 35, i to je iznos naše varijable *delay*, za slučaj da želimo otkucati 60 slova u minuti.

```
#include <12F683.h>              // define device
#include <stdlib.h>              // standard library
#include <string.h>              // string library
#include <float.h>               // floating point operations
#fuses NOWDT,NOPROTECT,INTRC_IO  // define config settings
#use delay(clock=4000000)        // define clock

// definiramo varijable
#DEFINE CW_KEYER PIN_A1          // zujalica
#DEFINE NE_DIRAJ  PIN_A3          // MCLR, ne koristit za IO
#DEFINE ONE_WIRE_PIN PIN_A0       // DS1820 is attached to PIN_A0
int delay;        // za telegrafiju, definiramo globalno

// funkcija za o(ne)mogućavanje prekida za vrijeme rada sa sondom
void onewire_disable_interrupts(int disable) {
   if (disable)
     disable_interrupts(GLOBAL);
   else
     enable_interrupts(GLOBAL);
}

// glavna funkcija
void main(void)
{
int a;
char output[36];
float precise;
// PIN_A0=DS18S20, PIN_A1=CW_KEYER, PIN_A2=NC
// PIN_A3=NC, PIN_A4=NC, PIN_A5=NC

delay=50;

for (a=0;a<36;a++) { output[a]=' '; } // čišćenje stringa

while (1)
{
 if (onewire_init_with_error_check())
  {
   precise=onewire_ds1820_read_temp_c_extended_float();
   // zapisem poruku i temperaturu
   sprintf(output,"9A0BVP JN65XF %3.0fC WWW.RKP.HR/9A0BVP",precise);
  }
else
  {
   // otkuca se samo poruka ako temperatura nije dostupna
   strcpy(output,"9A0BVP JN65XF WWW.RKP.HR/9A0BVP ");
  }

 for (a=0;a<36;a++)
  {
   cw_send(output[a]);
  }

 // drzin key down
 key_down();
}
```

Sklop se uključuje uključivanjem napajanja. Ukoliko je potrebno, može se dodati i sklopka.