METROPOLITNÍ
UNIVERZITA PRAHA

Technologicko-experimentální praxe

Arduino (duben 2018)

doc. Ing. Tomáš Frýza, Ph.D.

Obsah

1 Základní použití platformy Arduino **4**
1.1 Arduino Uno 4
1.2 Asynchronní komunikace 7

2 Řízení sériové komunikace I2C **11**

3 Komunikace s WiFi modulem **17**

Seznam obrázků

1.1 Vývojová deska Arduino Uno 4
1.2 Vývojové prostředí Arduino IDE s příkladem `Blink` 5
1.3 Časový průběh signálu LED diody 6
1.4 Zapojení druhé LED diody prostřednictvím nepájivého pole 7
1.5 Průběh časového signálu asynchronní komunikace s dekódovanými daty 9
2.1 Zachycení sériové komunikace na sběrnici I2C 14
3.1 Označení pinů na modulu ESP8266 ESP-01 18
3.2 Vizualizace měřených dat na serveru `https://thingspeak.com/` 21

Seznam tabulek

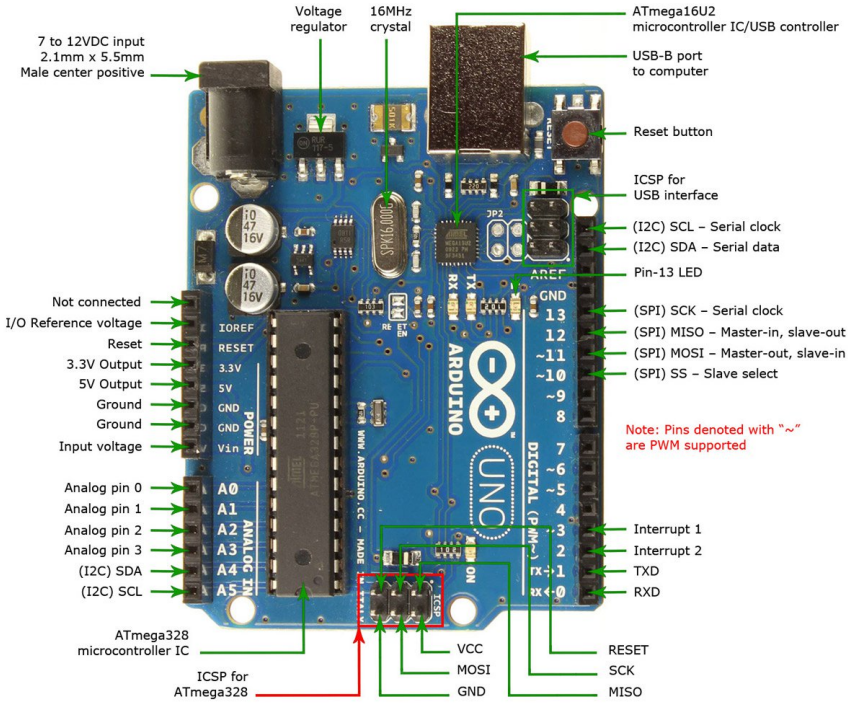
2.1 Uložení dat v kombinovaném senzoru DHT12 12
2.2 Uložení dat v obvodu reálného času RTC DS3231 14

1 Základní použití platformy Arduino

Cílem tohoto cvičení je seznámení s populární platformou Arduino, která obsahuje 8bitový mikrokontrolér AVR a je vhodná pro celou řadu řídicích aplikací. Aplikace budou programovány v jazyce C++ v prostředí Arduino IDE, které je zdarma dostupné pro Windows, Mac OS X i Linux. Pro verifikaci správné funkce aplikací bude použit logický analyzátor firmy Seleae, Inc.

1.1 Arduino Uno

Arduino je projekt vyvíjející otevřené platformy založené na 8bitovém mikrokontroléru s architekturou AVR. Podrobný popis HW modulů, včetně dostupných SW nástrojů, knihoven a manuálů naleznete na webových stránkách http://www.arduino.cc. Ve cvičení je využívána základní vývojová deska Arduino Uno, resp. její levnější, ale plně kompatibilní čínský klon.



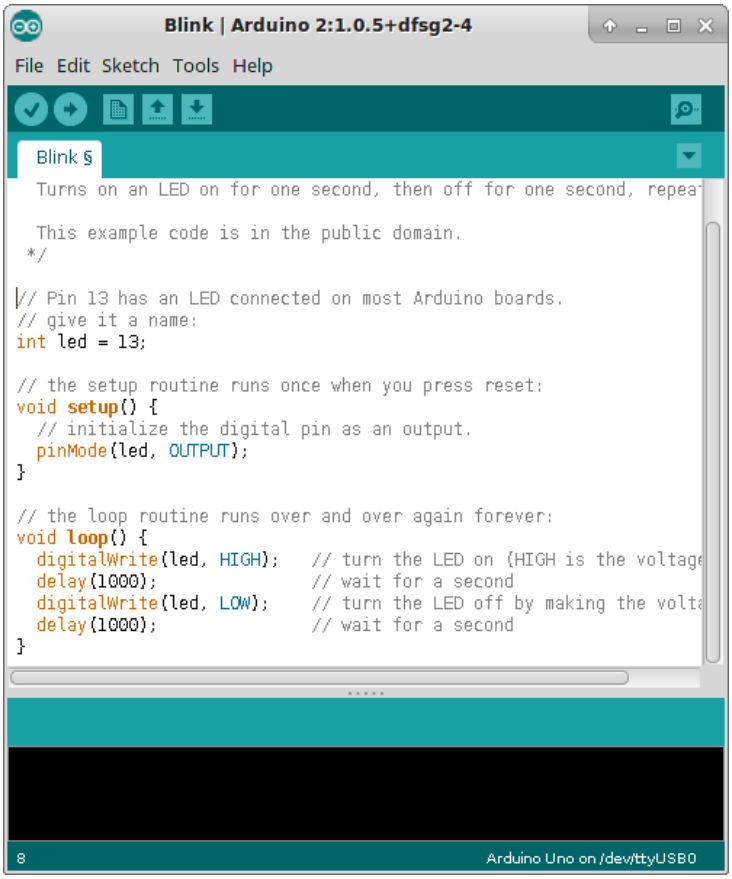
Obrázek 1.1: Vývojová deska Arduino Uno

Vývojové prostředí Arduino IDE je velmi jednoduché a kromě textového editoru nabízí několik ovládacích prvků. My využijeme pouze: `Verify` (Compile) pro přeložení zdrojového kódu, `Upload` pro nahrání binární verze kódu do vývojové desky a `Serial Monitor` k otevření nového okna s asynchronní komunikaci mezi mikrokontrolérem a počítačem po USB kabelu.

Příklad 1.1. *Naprogramujte aplikaci pro blikání jedné LED diody.*

Řešení 1.1. Prostředí obsahuje také základní sadu příkladů. V menu `File / Examples / 01.Basics` vyberte příklad `Blink`. Příklad se ovevře v novém okně, viz obrázek 1.2.

Zdrojové kódy pro Arduino svou strukturou usnadňují vývoj aplikací i méně zkušeným programátorům. Standardně obsahují dvě funkce: `setup()`, která se spustí jedenkrát po resetu mikrokontroléru a `loop()`, která představuje nekonečnou smyčku hlavní aplikace a neustále se opakuje.



Obrázek 1.2: Vývojové prostředí Arduino IDE s příkladem `Blink`

V aplikaci je použita globální proměnná `led = 13`, která specifikuje jeden konkrétní pin mikrokontroléru. Právě na pinu číslo 13 je na Arduino Uno připojena LED dioda. Dále příklad volá jen tři funkce: `pinMode()`, pomocí které se konfiguruje náš pin jako výstupní, `digitalWrite()` je funkce pro nastavení vysoké nebo nízké napěťové úrovně na výstupním pinu a `delay()`, která pozdrží výkon programu na stanovený počet milisekund.

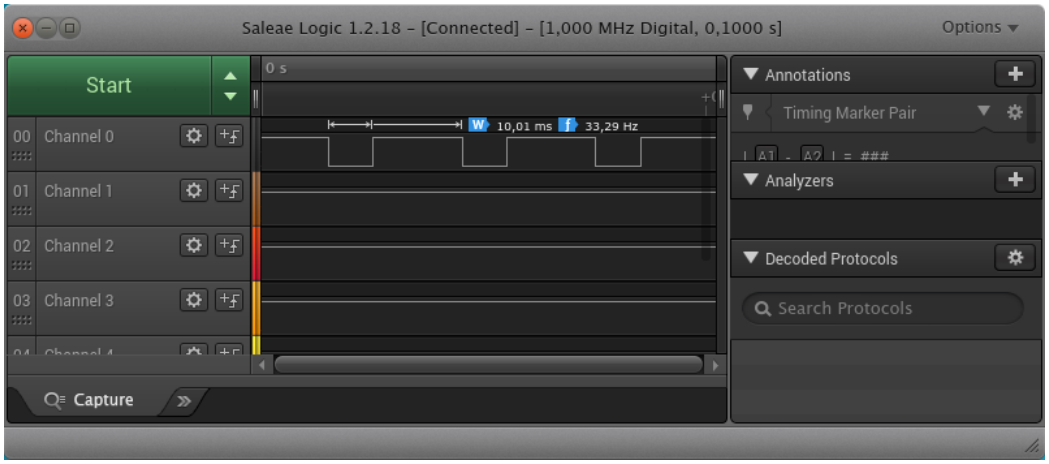
```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Otestujte si svůj zrak a postupně snižujte hodnotu zpoždění až nebude pro vás blikání LED na desce pozorovatelné.

Příklad 1.2. *Ověřte dobu zpoždění pomocí logického analyzátoru.*

Řešení 1.2. Alogický analyzátor je HW zařízení, které je schopné převádět reálné signály do číslicové podoby a zobrazovat jejich časové průběhy. Ve cvičení bude využit levný klon analyzátoru firmy Saelea, Inc., jehož ovládací a zobrazovací software je zdarma stažitelný z webových stránek https://www.saleae.com/downloads pro libovolnou platformu.

Pomocí dvou vodičů připojte GND vstup analyzátoru s GND pinem na Arduino desce a vstup CH1 s pinem 13. Spusťte aplikaci analyzátoru `Logic` a s pomocí zeleného tlačítka se dvěma šipkami nastavte dobu snímání vstupního signálu `Duration (Record data for)` na 100 milisekund a rychlost vzorkování `Speed (Sample Rate)` na 1 MS/s. Tlačítkem `Start` spustíte vzorkování (záznam) signálu pro LED diodu na pinu 13. Ověřte správnost časování generovaného signálu pro odlišné hodnoty zpoždění funkce `delay()`.

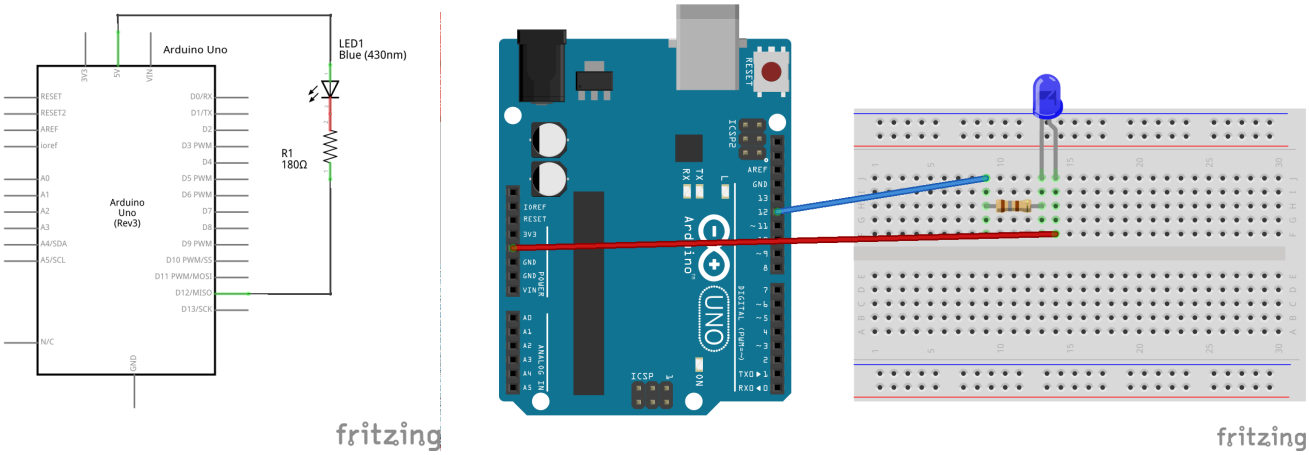


Obrázek 1.3: Časový průběh signálu LED diody. (Doba zpoždění HIGH zde nastavena na 20 a LOW na 10 ms.)

Příklad 1.3. *Přidejte do aplikace druhou LED a zajistěte, aby blikaly střídavě.*

Řešení 1.3. Pomocí nepájivého pole zapojte druhou LED diodu a rezistor omezující elektrický proud jí tekoucí mezi pin 12 a napájecí napětí 5 V dle obrázku 1.4. Do zdrojového kódu přidejte řádky ovládající výstupní pin 12. Celá aplikace může vypadat dle následujícího výpisu.

```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;
int led2 = 12;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
  pinMode(led2, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
```



Obrázek 1.4: Zapojení druhé LED diody prostřednictvím nepájivého pole

```
  digitalWrite(led2, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(250);                 // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  digitalWrite(led2, LOW);    // turn the LED off by making the voltage LOW
  delay(500);                 // wait for a second
}
```

Všimněte si, že obě LED diody svítí při odlišných úrovních signálů na pinech 12 a 13. Je to dáno orientací LED diod: zelená LED na vývojové desce je připojena anodou na pin 13 a katodou na GND, zatímco LED na nepájivém poli má anodu na +5 V a katodu na řídicí pin 12.

1.2 Asynchronní komunikace

Příklad 1.4. *Vytvořte aplikaci vysílající asynchronní data z Arduina do počítače.*

Řešení 1.4. Otevřete si příklad z menu `File / Examples / 04.Communication / ASCIITable` využívající asynchronní jednosměrné komunikace od vývojové desky do počítače. Po překompilování kódu a nahrání do mikrokontroléru na desce, spusťte také `Serial Monitor` (ikona vpravo nahoře). Do nově otevřeného okna se postupně vypíší zobrazitelné znaky z ASCII tabulky a jejich kódy v desítkové, šestnáctkové, osmičkové a binární soustavě.

V aplikaci jsou využity funkce pro konfiguraci vysílače sériových dat `begin(9600)` se symbolovou rychlostí 9 600 Bd, dále dvě funkce pro vysílání textových řetězců po asynchronní sériové lince `println()` a `print()` a vyslání jednoho sériového rámce funkcí `write()`. S pomocí manuálu na webových stránkách https://www.arduino.cc/reference/en/language/functions/communication/serial/ příp. experimentálně zjistěte význam těchto funkcí.

```
void setup() {
 //Initialize serial and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }

  // prints title with ending line break
  Serial.println("ASCII Table ~ Character Map");
}

// first visible ASCIIcharacter '!' is number 33:
```

```
int thisByte = 33;
// you can also write ASCII characters in single quotes.
// for example. '!' is the same as 33, so you could also use this:
//int thisByte = '!';

void loop() {
  // prints value unaltered, i.e. the raw binary version of the
  // byte. The serial monitor interprets all bytes as
  // ASCII, so 33, the first number,  will show up as '!'
  Serial.write(thisByte);

  Serial.print(", dec: ");
  // prints value as string as an ASCII-encoded decimal (base 10).
  // Decimal is the  default format for Serial.print() and Serial.println(),
  // so no modifier is needed:
  Serial.print(thisByte);
  // But you can declare the modifier for decimal if you want to.
  //this also works if you uncomment it:

  // Serial.print(thisByte, DEC);


  Serial.print(", hex: ");
  // prints value as string in hexadecimal (base 16):
  Serial.print(thisByte, HEX);

  Serial.print(", oct: ");
  // prints value as string in octal (base 8);
  Serial.print(thisByte, OCT);

  Serial.print(", bin: ");
  // prints value as string in binary (base 2)
  // also prints ending line break:
  Serial.println(thisByte, BIN);

  // if printed last visible character '~' or 126, stop:
  if(thisByte == 126) {     // you could also use if (thisByte == '~') {
    // This loop loops forever and does nothing
    while(true) {
      continue;
    }
  }
  // go on to the next character
  thisByte++;
}
```

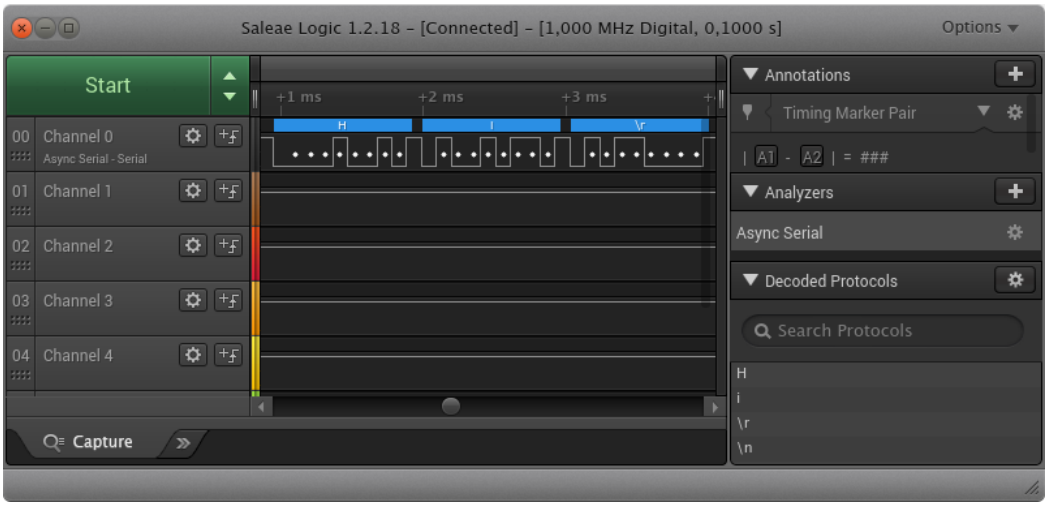
Příklad 1.5. *Ověřte strukturu asynchronních rámců.*

Řešení 1.5. Ukázkový kód z předchozího příkladu zjednodušte na vyslání krátkého řetězce a logickým analyzátorem připojeným na pin `TX->1` zachyťte časový průběh takovéto komunikace. Ověřte, že (a) komunikace se zahajuje start bitem, (b) počet datových bitů je osm a jsou seřazeny od LSB k MSB, (c) není použita žádná parita a (d) počet stop bitů je jeden.

```
void setup() {
 //Initialize serial and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
```

```
  }
}

void loop() {
  Serial.println("Hi");
  delay(100);
}
```

Logický analyzátor lze také nastavit pro přehlednější dekódování sériových dat. Stiskněte tlačítko "+"v části `Analyzers` a vyberte typ komunikace `Async Serial`. Ponechte defaultní nastavení komunikačního rámce a stiskněte tlačítko `Save`. Do časového průběhu se zobrazí dekódované hodnoty jednotlivých komunikačních rámců.



Obrázek 1.5: Průběh časového signálu asynchronní komunikace s dekódovanými daty

S pomocí analyzátoru experimentálně zjistěte rozdíl v použití vysílacích funkcí `print()` a `println()`.

Příklad 1.6. *Vytvořte aplikaci kombinující vysílač i přijímač asynchronních dat.*

Řešení 1.6. Kombinací kódů s LED diodou a vysílače sériových dat vznikne následující aplikace. V nekonečné smyčce je navíc volána funkce `parseInt()`, která reprezentuje přijímač sériových dat a plní globální proměnnou `delayParam` (popis celé funkce viz https://www.arduino.cc/reference/en/language/functions/communication/serial/parseint/). Tato proměnná udává dobu zpoždění pro blikání LED diodou.

```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;
int delayParam = 1000;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);

 //Initialize serial and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
```

```
    ; // wait for serial port to connect. Needed for Leonardo only
  }
}

// the loop routine runs over and over again forever:
void loop() {
  // check if data has been sent from the computer:
  if (Serial.available()) {
    // read the most recent integer:
    delayParam = Serial.parseInt();
  }

  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  Serial.println("HIGH");
  Serial.println(delayParam);
  delay(delayParam);                // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  Serial.println("LOW");
  delay(delayParam);                // wait for a second
}
```

2 Řízení sériové komunikace I2C

Cílem tohoto cvičení je seznámení se sériovou sběrnicí I2C (Inter-integrated Circuit). Sériová sběrnice obsahuje dva vodiče: pro přenos dat s označením SDA a pro přenos hodinových impulzů SCL a umožňuje snadné propojení jednoho nadřízeného obvodu (tzv. master) s více podřízenými obvody (slave). Jako master je použit mikrokontrolér AVR na vývojové desce Arduino Uno a slave obvody jsou kombinované teplotní a vlhkostní čidlo DHT12 a obvod reálného času DS3231. Aplikace budou programovány v jazyce C++ v prostředí Arduino IDE. Pro verifikaci správné funkce aplikací bude použit logický analyzátor firmy Saleae, Inc.

Příklad 2.1. *Naprogramujte aplikaci pro skenování sběrnice I2C.*

Řešení 2.1. Pomocí nepájivého pole připojte na datový (SDA) a hodinový (SCL) signál sběrnice I2C moduly pro měření teploty/vlhkosti DHT12 a obvod reálného času DS3231. Pro napájení obou modulů použijte napětí 3.3 V a GND z vývojové desky Arduina.

Úkolem skenovací aplikace je vyslat postupně všechny kombinace adres na I2C (adresa je 7bitová, tj. hodnoty od 0 do 127) a "poslouchat", zda-li některý z obvodů na tyto adresy neza-reaguje. Využijte příklad z prostředí Arduino IDE, který naleznete v menu `File / Examples / Wire / SFRRanger_reader` a upravte ho do následující podoby.

```
#include <Wire.h>

void setup()
{
  Wire.begin();                // join i2c bus (address optional for master)
  Serial.begin(9600);          // start serial communication at 9600bps
  Serial.println("---I2C SCANNER---");
}

int addr = 0;

void loop()
{
  for (addr=0; addr<128; addr++) {
    Serial.println(addr);

    Wire.requestFrom(addr, 1);      // request 1 bytes from slave device #addr

    if (1 <= Wire.available()) {    // if one bytes is available for reading
      Serial.print("...OK...$");
      Serial.println(addr, HEX);
    }
    delay(100);
  }

  while(1) {
    ;
  }
}
```

Dokumentace k jednotlivým funkcím knihovny `wire` je dostupná na internetových stránkách https://www.arduino.cc/en/Reference/Wire. Aplikaci zkompilujte (`Verify`), nahrajte do Arduino Uno (Upload) a otevřete monitor sériové asynchronní komunikace, kterou aplikace využívá (`Serial Monitor`). Pozn.: Ve funkci `setup()` je nastavena komunikace se symbolovou rychlostí 9 600 Bd; ujistěte se, že konzole sériového monitoru má nastavenou tutéž rychlost.

Z textového výstupu aplikace je patrné, že na sběrnici jsou připojeny celkem tři obvody s I2C adresou. Z internetu, či katalogových listů lze vyčíst, že `0x57` odpovídá sériové paměti EEPROM (umístěné na modulu RTC), `0x5c` je kombinovaný teplotní/vlhkostní senzor a adresu `0x68` obsahuje obvod RTC.

```
...
86
87
...OK...$57
88
89
90
91
92
...OK...$5C
93
94
95
96
97
98
99
100
101
102
103
104
...OK...$68
105
106
...
```

Příklad 2.2. *Naprogramujte aplikaci komunikující s kombinovaným teplotním a vlhkostním čidlem DHT12.*

Řešení 2.2. Zapojení aplikace neměňte. Opět můžete vyjít z ukázky `File / Examples / Wire / SFRRanger_reader`, kterou upravte pro vyčítání prvních pěti bytů z obvodu DHT12. Pozn.: Z manuálu kombinovaného čidla lze vyčíst, že data jsou v něm uložena dle tabulky 2.1.

Tabulka 2.1: Uložení dat v kombinovaném senzoru DHT12

Adresa	Popis dat
0x00	Celá část relativní vlhkosti
0x01	Desetinná část relativní vlhkosti
0x02	Celá část teploty
0x03	Desetinná část teploty
0x04	Kontrolní součet všech dat

```
#include <Wire.h>

void setup()
{
  Wire.begin();                // join i2c bus (address optional for master)
```

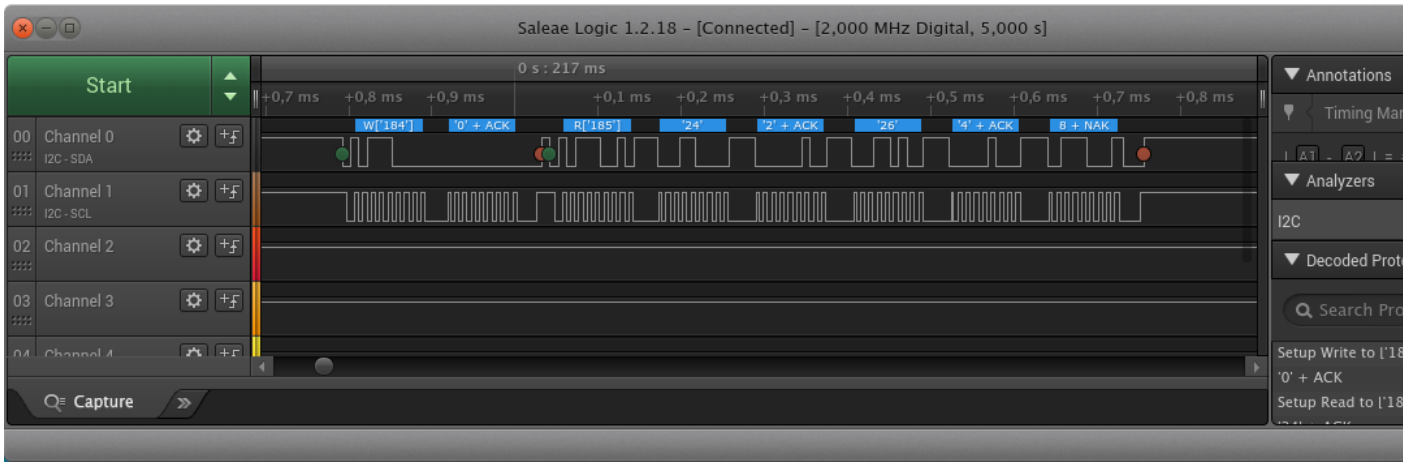
```
  Serial.begin(9600);           // start serial communication at 9600bps
  Serial.println("---DHT12 TEST PROGRAM---");
}

unsigned char humd = 0;
unsigned char temp = 0;
float temperature = 0.0;
unsigned char check = 0;

void loop()
{
  Wire.beginTransmission(0x5c); // transmit to temp/humid. device only
  Wire.write(byte(0x00));       // sets register pointer to register 0x00
  Wire.endTransmission();       // stop transmitting

  Wire.requestFrom(0x5c, 5);    // request 5 bytes from slave device 0x5c

  if(5 <= Wire.available())     // if 5 bytes were received
  {
    Serial.print("Humidity: ");
    humd = Wire.read();         // read byte from addr. 0x00
    Serial.print(humd);         // display data
    Serial.print(".");
    humd = Wire.read();         // read byte from addr. 0x01
    Serial.print(humd);         // display data
    Serial.println(" %RH");

    Serial.print("Temperature: ");
    temp = Wire.read();         // read byte from addr. 0x02
    Serial.print(temp);
    Serial.print(".");
    temperature = (float)temp;
    temp = Wire.read();         // read byte from addr. 0x03
    Serial.print(temp);
    Serial.println(" deg");
    temperature = temperature + ((float)temp/10);
    Serial.println(temperature);

    check = Wire.read();        // read byte from addr. 0x04
    Serial.print("Checksum: ");
    Serial.println(check, BIN);
    Serial.println();
  }

  delay(5000);
}
```

Textový výstup spuštěné aplikace může být následující:

```
---DHT12 TEST PROGRAM---
Humidity: 25.7 %RH
Temperature: 26.4 deg
26.40
Checksum: 111110

Humidity: 25.2 %RH
Temperature: 26.4 deg
26.40
Checksum: 111001
```

Příklad 2.3. *Ověřte komunikační protokol I2C a strukturu jednotlivých rámců pomocí logického analyzátoru.*

Řešení 2.3. Na sběrnici I2C připojte dva kanály logického analyzátoru Saleae (používaném v předchozím cvičení) a spusťte vizualizační program `Logic` na počítači. Zachyťte komunikaci na sběrnici, nastavte si dekodér I2C v části `Analyzers` a ověřte posloupnost adresních a datových paketů na sběrnici.



Obrázek 2.1: Zachycení sériové komunikace na sběrnici I2C

Z obrázku 2.1 je patrné, že datová linka adresuje obvod nejprve pro zápis s následným datovým slovem `0x00` a po opětovném startu komunikace adresuje pro čtení, kdy se vyčte pět datových paketů dle tabulky 2.1.

Příklad 2.4. *Naprogramujte aplikaci komunikující s obvodem reálného času DS3231.*

Řešení 2.4. Aplikaci z bodu 2.2 rozšiřte o funkci komunikující z obvodem RTC (I2C adresa `0x68`). Dle manuálu k obvodu DS3231 lze mj. zjistit umístění dat odpovídající sekundám a minutám, viz tabulka 2.2.

Tabulka 2.2: Uložení dat v obvodu reálného času RTC DS3231

Adresa	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Popis
0x00	0	Desítky sekund			Jednotky sekund				Sekundy
0x01	0	Desítky minut			Jednotky minut				Minuty
0x02	...								

```
#include <Wire.h>

void setup()
{
  Wire.begin();                 // join i2c bus (address optional for master)
  Serial.begin(9600);           // start serial communication at 9600bps
  Serial.println("---TEMP + RTC test---");
}
```

```
unsigned char temp = 0;
float temperature = 0.0;
unsigned char mins10 = 0;
unsigned char mins1 = 0;
unsigned char secs10 = 0;
unsigned char secs1 = 0;


void loop()
{
  getRTC();
  getTemp();
  Serial.println();
  delay(5000);
}


void getRTC()
{
  Wire.beginTransmission(0x68);// transmit to rtc device only
  Wire.write(byte(0x00));      // sets register pointer to register 0x00
  Wire.endTransmission();      // stop transmitting

  Wire.requestFrom(0x68, 2);   // request 2 bytes from slave device 0x68

  if (2 <= Wire.available()) {   // if 2 bytes were received
    Serial.print("Time: ");
    temp = Wire.read();
    secs10 = temp >> 4;          // get 10 Seconds
    secs1 = temp & 0x0f;         // get Seconds
    temp = Wire.read();
    mins10 = temp >> 4;          // get 10 Minutes
    mins1 = temp & 0x0f;         // get Minutes

    Serial.print(mins10);
    Serial.print(mins1);
    Serial.print(":");
    Serial.print(secs10);
    Serial.println(secs1);
  }
}


void getTemp()
{
  Wire.beginTransmission(0x5c);// transmit to temp/humid. device only
  Wire.write(byte(0x02));      // sets register pointer to register 0x02
  Wire.endTransmission();      // stop transmitting

  Wire.requestFrom(0x5c, 2);   // request 2 bytes from slave device 0x5c

  if (2 <= Wire.available()) {   // if 2 bytes were received
    Serial.print("Temperature: ");
    temp = Wire.read();
    temperature = (float)temp;
    temp = Wire.read();
    temperature = temperature + ((float)temp/10);
    Serial.print(temperature);
    Serial.println(" deg");
  }
}
```

Textový výstup aplikace může být následující:

```
---TEMP + RTC test---
Time: 35:08
Temperature: 26.40 deg

Time: 35:13
Temperature: 26.40 deg
```

3 Komunikace s WiFi modulem

Cílem tohoto cvičení je aplikovat znalosti z předchozích cvičení a vytvořit jednoduchý bod v síti IoT (Internet of Things). Pro bezdrátovou komunikaci je použit WiFi modul ESP8266 ESP-01, dále kombinované teplotní a vlhkostní čidlo DHT12 a pro sběr a vizualizaci dat server https://thingspeak.com/ firmy MathWorks, Inc. Aplikace budou programovány v jazyce C++ v prostředí Arduino IDE. Pro verifikaci správné komunikace s WiFi modulem může být použit logický analyzátor firmy Saleae, Inc.

Příklad 3.1. *Naprogramujte aplikaci pro komunikaci s kombinovaným senzorem DHT12.*

Řešení 3.1. Pomocí nepájivého pole připojte na datový (SDA) a hodinový (SCL) signál sběrnice I2C modul pro měření teploty/vlhkosti DHT12. Pro napájení modulu použijte napětí 3.3 V a GND z vývojové desky Arduina.

Použijte kód z předešlého cvičení a z opakující se funkce `loop()` volejte funkce pro měření teploty `getTemp()` a vlhkosti `getHumd()`, které naplní čtyři globální proměnné s aktuální teplotou a relativní vlhkostí (vždy celá a desetinná část hodnoty, viz tabulka 2.1).

```
#include <Wire.h>

void setup()
{
  Wire.begin();
}

unsigned char temp1 = 0;
unsigned char temp2 = 0;
unsigned char humd1 = 0;
unsigned char humd2 = 0;

void loop()
{
  getTemp();
  getHumd();
  delay(60000);
}

void getTemp()
{
  Wire.beginTransmission(0x5c);// transmit to temp/humid. device only
  Wire.write(byte(0x02));      // sets register pointer to register 0x02
  Wire.endTransmission();      // stop transmitting

  Wire.requestFrom(0x5c, 2);   // request 2 bytes from slave device 0x5c

  if (2 <= Wire.available()) { // if 2 bytes were received
    temp1 = Wire.read();
    temp2 = Wire.read();
  }
}


void getHumd()
{
  Wire.beginTransmission(0x5c);// transmit to temp/humid. device only
  Wire.write(byte(0x00));      // sets register pointer to register 0x00
  Wire.endTransmission();      // stop transmitting
```

```
  Wire.requestFrom(0x5c, 2);    // request 2 bytes from slave device 0x5c

  if (2 <= Wire.available()) { // if 2 bytes were received
    humd1 = Wire.read();
    humd2 = Wire.read();
  }
}
```

Senzor komunikuje prostřednictvím dvouvodičové sběrnice I2C. Dokumentace k jednotlivým funkcím knihovny `wire` je dostupná na internetových stránkách https://www.arduino.cc/en/Reference/Wire a byly vysvětleny na předchozím cvičení.

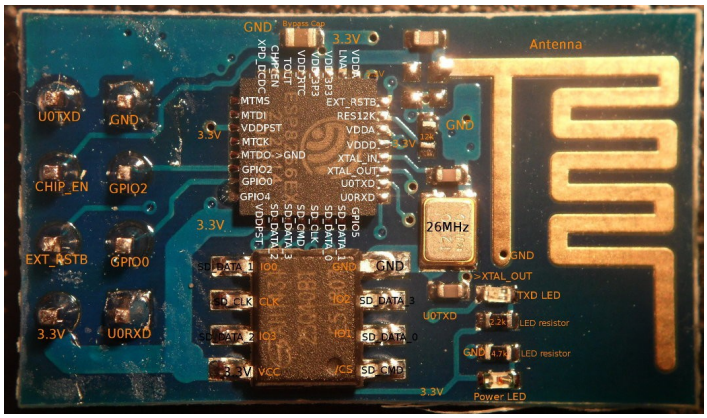
Příklad 3.2. *Vytvořte účet na serveru pro vizualizaci dat.*

Řešení 3.2. Postupujte dle následujících pokynů:

1. Na serveru https://thingspeak.com/ vytvořte zdarma účet pomocí tlačítka `Get Started For Free`.
2. Vyplňte své kontaktní údaje a email, který používáte; bude na něj zasláno potvrzení pro zřízení účtu.
3. Vytvořte nový komunikační kanál `New Channel`, vyplňte `Name` a případný popis `Description` a změňte názvy `Field 1` a `Field 2` na `Temperature` a `Humidity`.
4. Data budou zobrazována v záložce `Private View`; v záložce `API Keys` si zkopírujte textový řetězec `Write API Key`, který bude sloužit jako identifikátor vašeho kanálu během WiFi komunikace.

Příklad 3.3. *Pomocí WiFi modulu ESP8266 ESP-01 odesílejte data se server.*

Řešení 3.3. Pomocí nepájivého pole připojte WiFi modul ESP8266 ESP-01 a Arduino Uno dle obrázku 3.1. Výměna dat mezi modulem a řídicím mikrokontrolérem probíhá prostřednictvím asynchronní komunikace. Všimněte si, že vysílací pin modulu je proto spojen s přijímacím pinem Arduino a obráceně.



ESP8266 ESP-01	Arduino Uno
U0TXD	Rx (pin 0)
CHIP_EN	3.3V
EXT_RSTB	Nepřipojeno
3.3V	3.3V
GND	GND
GPIO2	Nepřipojeno
GPIO0	Nepřipojeno
U0RXD	Tx (pin 1)

Obrázek 3.1: Označení pinů na modulu ESP8266 ESP-01

Do zdrojového kódu z příkladu 3.1 přidejte funkce pro konfiguraci WiFi modulu `wifi_config()` a pro opakované posílání dat `wifi_send()`.

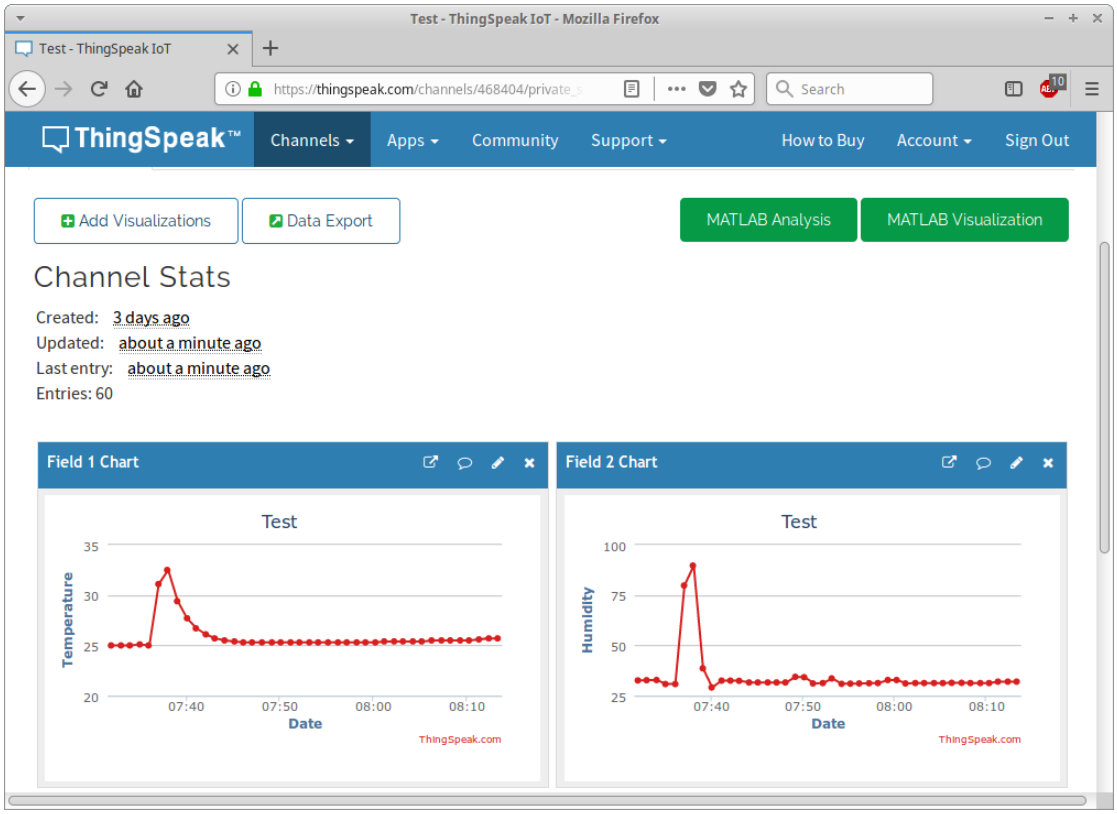
```
#include <Wire.h>        // I2C library

String ssid ="xxx";      // ssid of your wifi network
String password = "xxx"; // password of your wifi network
String ip_addr = "api.thingspeak.com";
String api_key = "xxx";  // paste your Write API Key here

void setup() {
  Wire.begin();          // setup I2C communication
  Serial.begin(115200);  // setup async. communication with 115200 speed
  wifi_config();         // setup wifi modul and get ip address from your router
}

unsigned char temp1 = 0; // two bytes for temperature
unsigned char temp2 = 0;
unsigned char humd1 = 0; // two bytes for humidity
unsigned char humd2 = 0;

void loop() {
  getTemp();             // get temperature via I2C
  getHumd();             // get humidity via I2C
  wifi_send();           // send both values via async. communication
  delay(60000);          // wait for 1 min
}

void getTemp()
{
  Wire.beginTransmission(0x5c);// transmit to temp/humid. device only
  Wire.write(byte(0x02));      // sets register pointer to register 0x02
  Wire.endTransmission();      // stop transmitting

  Wire.requestFrom(0x5c, 2);   // request 2 bytes from slave device 0x5c

  if (2 <= Wire.available()) { // if 2 bytes were received
    temp1 = Wire.read();
    temp2 = Wire.read();
  }
}

void getHumd()
{
  Wire.beginTransmission(0x5c);// transmit to temp/humid. device only
  Wire.write(byte(0x00));      // sets register pointer to register 0x00
  Wire.endTransmission();      // stop transmitting

  Wire.requestFrom(0x5c, 2);   // request 2 bytes from slave device 0x5c

  if (2 <= Wire.available()) { // if 2 bytes were received
    humd1 = Wire.read();
    humd2 = Wire.read();
  }
}

void wifi_config() {
  Serial.println("AT");  // test wifi module
  delay(1000);
  Serial.println("AT+CWMODE=1");  // set mode to STA
  delay(2000);
  String cmd = "AT+CWJAP=\"" +ssid+"\",\"" + password + "\"";
```

```
  Serial.println(cmd);   // login to wifi network and get ip address
  delay(5000);
}

void wifi_send() {
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += ip_addr;
  cmd += "\",80";
  Serial.println(cmd);           // start comunication with server
  delay(500);
  cmd = "GET /update?api_key=";
  cmd += api_key;                // prepare request including measured values
  cmd += "&field1=";
  cmd += String(temp1);          // include temperature
  cmd += ".";
  cmd += String(temp2);
  cmd += "&field2=";
  cmd += String(humd1);          // include humidity
  cmd += ".";
  cmd += String(humd2);
  cmd += "\r\n";
  Serial.print("AT+CIPSEND=");
  Serial.println(cmd.length());  // send number of bytes first
  delay(100);
  Serial.print(cmd);             // send request including measured values
}
```

Vzhledem k tomu, že programování Arduina probíhá na stejných pinech, kde je připojen i WiFi modul, před samotným uploadem kódu odpojte vodiče z pinů 0 a 1 (Rx a Tx). Po úspěšném naprogramování vodiče opět připojte. Spusťte `Serial Monitor` na kterém pozorujte část komunikace s WiFi modulem (pozor, odpovědi modulu se nezobrazují):

```
AT
AT+CWMODE=1
AT+CWJAP="xxx","xxx"
AT+CIPSTART="TCP","api.thingspeak.com",80
AT+CIPSEND=62
GET /update?api_key=xxx&field1=25.0&field2=31.3
AT+CIPSTART="TCP","api.thingspeak.com",80
AT+CIPSEND=62
```

Je-li komunikace prostřednictvím WiFi sítě korektní, na webové stránce `thingspeak.com` se začnou shromažďovat a zobrazovat změřená data, viz obrázek 3.2. Pokud tomu tak není, zkontrolujte zda v kódu nemáte chybu, nebo si zobrazte asynchronní komunikaci na pinech 0 a 1 pomocí logického analyzátoru.



Obrázek 3.2: Vizualizace měřených dat na serveru https://thingspeak.com/