1496818942
VÝROBNÍ ČÍSLO

Čtečka otisků prstů

1. POPIS

Jedná se o modul, který lze po menších úpravách připojit k Arduinu (viz. kapitola Zapojení). Čtečka otisků prstů disponuje vlastní pamětí, takže si pamatuje otisky prstů, které do ní uživatel uloží. Jedná se o vhodné řešení pro identifikaci osob, které lze využít zejména v bezpečnostních zařízeních nebo například docházkových systémech atd. Zařízení je chráněno černým plastovým krytem s průhlednou plochou pro vložení prstu.

Základní charakteristika:

- Kompaktní rozměry
- Paměť otisků prstů (až 162 otisků)
- Spolehlivé snímání
- Řešení pro inicializaci osob bezpečnostními systémy atd.

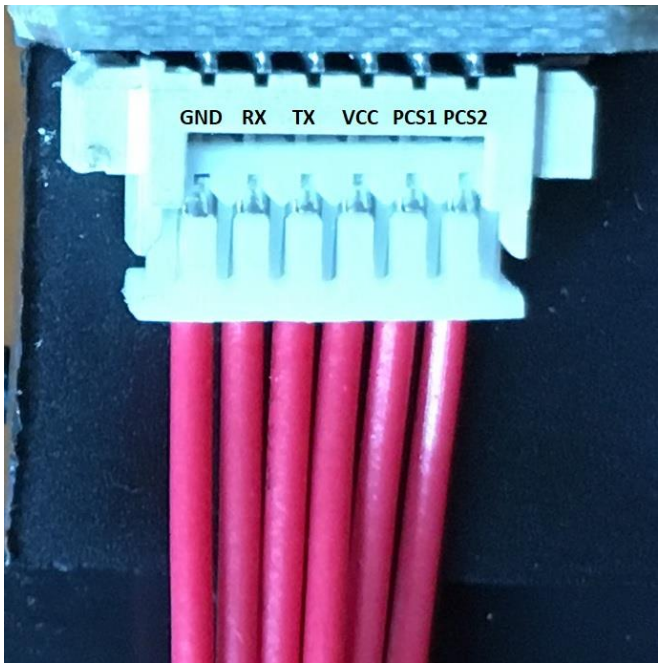
2. SPECIFIKACE

Napájení	3,3 nebo 5 V	**Interface**	UART
Proud	< 65 mA	**Baud rate**	57600 bps
Špičkový proud	< 95 mA	**Pracovní teplota**	-20 až 50 °C
Max. počet uložených otisků	162	**Pracovní vlhkost**	40 až 85 % RH
Odezva snímání	< 1 s	**Rozměry (mm)**	56 x 20 x 21,5

3. Zapojení

Poznámka:

U konkrétního modelu jsou všechny vodiče červené barvy. Je tedy nutné, aby uživatel dodržel pořadí pinů viz. obrázek níže. Na druhém konci kabeláže nejsou standardní piny, které je možné zasunout do Arduina. Druhý konec je osazen stejným konektorem, jímž jsou vodiče připojeny do snímače otisků prstů. Pro připojení do Arduina doporučujeme konektor ustřihnout a poté vodiče buď osadit vhodnými DuPont konektory (vyžadováno použití krimpovacích kleští) nebo alespoň potáhnout vrstvou cínu, aby byly v pinech Arduina pevně uchyceny a nevypadávaly.



Pinout

- **GND** → uzemnění
- **RX** → vysílaný signál – Arduino UNO pin 3
- **TX** → přijímaný signál – Arduino UNO pin 2
- **VCC** → napájení
- **PCS1** → bezkontaktní kapacitní senzor (v uvedeném příkladu není zapojen, umožňuje přesnější snímání – rychlá detekce prstu)
- **PCS2** → bezkontaktní kapacitní senzor (v uvedeném příkladu není zapojen, umožňuje přesnější snímání – rychlá detekce prstu)

4. UKÁZKA PROGRAMU

Pro chod tohoto programu je zapotřebí nainstalovat knihovny Adafruit_Fingerprint.h. Tento příklad (enroll) byl převzat z této knihovny a jeho funkcí je načíst otisky prstů do paměti s příslušným ID. Následná uložená data vyžadují ostatní příklady z této knihovny.

```
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>

uint8_t id;

uint8_t getFingerprintEnroll();

// Software serial for when you dont have a hardware serial port
// pin #2 is IN from sensor (GREEN wire)
// pin #3 is OUT from arduino  (WHITE wire)
// On Leonardo/Micro/Yun, use pins 8 & 9. On Mega, just grab a hardware serialport
SoftwareSerial mySerial(2, 3);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

// On Leonardo/Micro or others with hardware serial, use those! #0 is green wire, #1 is white
//Adafruit_Fingerprint finger = Adafruit_Fingerprint(&Serial1);


void setup()
{
  while (!Serial);  // For Yun/Leo/Micro/Zero/...
  delay(500);

  Serial.begin(9600);
  Serial.println("Adafruit Fingerprint sensor enrollment");

  // set the data rate for the sensor serial port
  finger.begin(57600);

  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1);
  }
}

uint8_t readnumber(void) {
  uint8_t num = 0;
```

```
  boolean validnum = false;
  while (1) {
    while (! Serial.available());
    char c = Serial.read();
    if (isdigit(c)) {
      num *= 10;
      num += c - '0';
      validnum = true;
    } else if (validnum) {
      return num;
    }
  }
}

void loop()                     // run over and over again
{
  Serial.println("Ready to enroll a fingerprint! Please Type in the ID # you want to save this finger as...");
  id = readnumber();
  Serial.print("Enrolling ID #");
  Serial.println(id);

  while (!  getFingerprintEnroll() );
}

uint8_t getFingerprintEnroll() {

  int p = -1;
  Serial.print("Waiting for valid finger to enroll as #"); Serial.println(id);
  while (p != FINGERPRINT_OK) {
    p = finger.getImage();
    switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println(".");
      break;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      break;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      break;
    default:
      Serial.println("Unknown error");
      break;
    }
  }
```

```
// OK success!

p = finger.image2Tz(1);
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

Serial.println("Remove finger");
delay(2000);
p = 0;
while (p != FINGERPRINT_NOFINGER) {
  p = finger.getImage();
}
Serial.print("ID "); Serial.println(id);
p = -1;
Serial.println("Place same finger again");
while (p != FINGERPRINT_OK) {
  p = finger.getImage();
  switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image taken");
    break;
  case FINGERPRINT_NOFINGER:
    Serial.print(".");
    break;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    break;
  case FINGERPRINT_IMAGEFAIL:
    Serial.println("Imaging error");
```

```
    break;
  default:
    Serial.println("Unknown error");
    break;
  }
}

// OK success!

p = finger.image2Tz(2);
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

// OK converted!
Serial.print("Creating model for #");  Serial.println(id);

p = finger.createModel();
if (p == FINGERPRINT_OK) {
  Serial.println("Prints matched!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  Serial.println("Communication error");
  return p;
} else if (p == FINGERPRINT_ENROLLMISMATCH) {
  Serial.println("Fingerprints did not match");
  return p;
} else {
  Serial.println("Unknown error");
  return p;
}
```

```
  Serial.print("ID "); Serial.println(id);
  p = finger.storeModel(id);
  if (p == FINGERPRINT_OK) {
    Serial.println("Stored!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_BADLOCATION) {
    Serial.println("Could not store in that location");
    return p;
  } else if (p == FINGERPRINT_FLASHERR) {
    Serial.println("Error writing to flash");
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }
}
```