Akademia Górniczo-Hutnicza
Wydział Informatyki, Elektroniki i Telekomunikacji

AGH

Złożone systemy cyfrowe
2017/2018

Temat projektu:

Programowanie myszki komputerowej

Autorzy:
Agnieszka Pierzchała
Kinga Kaczmarczyk

1. Założenia projektu

Zapoznanie się z zasadą działania portu szeregowego PS/2 oraz standardowej myszy komputerowej.

2. Funkcjonalność

Niskopoziomowe odczytywanie wartości z myszki.

3. Wymagany sprzęt

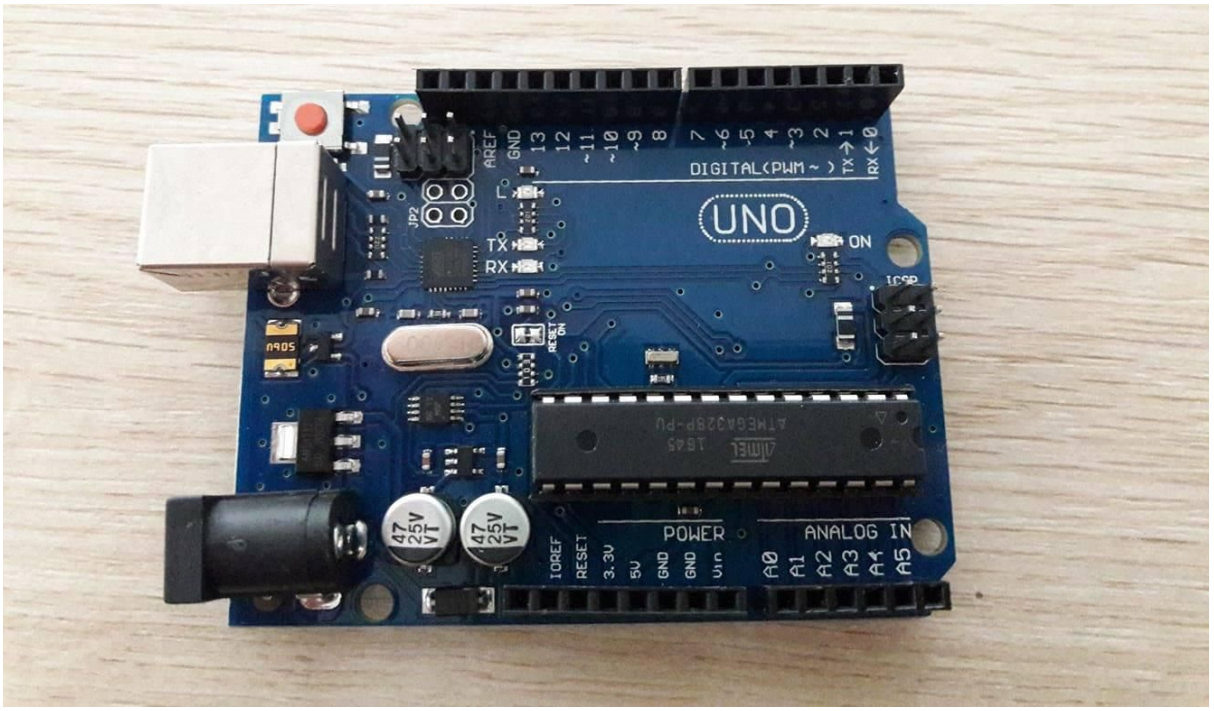
- Myszka komputerowa z kulką ze złączem żeńskim PS/2 Mini-DIN
- Płytka ARDUINO UNO.

4. Źródła

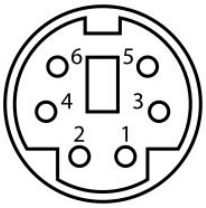
http://www.computer-engineering.org/ps2mouse/
https://www.arduino.cc/en/Reference.PinMode
https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite/

5. Etapy działania

a. Wybranie płytki sprzętowej



b. Podłączenie myszki do płytki Arduino poprzez łącze PS/2



Rozmieszczenie pinów

Pin	Nazwa	Funkcja
1	+DATA	Dane
2	Reserved	Zarezerwowane*
3	GND	Masa
4	Vcc	zasilanie +5V prądem stałym o natężeniu do 100mA
5	+CLK	Zegar
6	Reserved	Zarezerwowane**

Nazwa	Standard	Alternatywa	Alternatywa	Alternatywa	Alternatywa
+DATA	Biały	Żółty	Zielony	Szary	Zielony
GND	Żółty	Czarny	Pomarańczowy	Brązowy	Czerwony
Vcc	Czerwony	Czerwony	Niebieski	Czerwony	Czarny
+CLK	Zielony	Niebieski	Biały	Żółty	Biały

Podłączono złącze do płytki Arduino w następujący sposób:

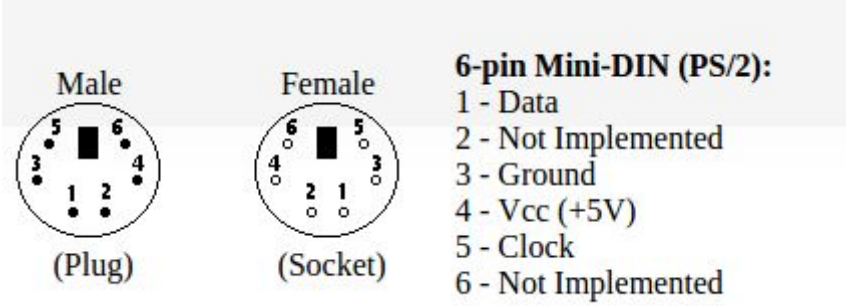
+5V to +5V of Arduino
Ground to GND of Arduino
Clock signal (CLK) to digital pin 6 of Arduino
Data to digital pin 5 of Arduino

Przy podłączeniu płytki do myszki przez łączę PS/2 zapoznano się z jego działaniem.

c. Interfejs PS/2

Zapoznanie się ze stroną fizyczną interfejsu PS/2

Fizyczny port PS / 2 jest jednym z dwóch rodzajów złączy: 5-pinowej DIN lub 6-pinowej mini-DIN. Oba złącza są (elektrycznie) podobne; jedyną praktyczną różnicą między nimi jest rozmieszczenie szpilek.



Myszka, której użyłyśmy do projektu miała złącze żeńskie Mini-DIN(PS/2)

Interfejs elektryczny łącza

Vcc / Ground zapewnia zasilanie klawiatury / myszy. Klawiatura lub mysz nie powinny pobierać więcej niż 275 mA od hosta.

Data i Clock są otwartymi kolektorami z rezystorami pullup do Vcc.Interfejs "otwarty kolektor" ma dwa możliwe stany: niskiej lub wysokiej impedancji.

Komunikacja

Mysz i klawiatura PS / 2 implementują dwukierunkowy synchroniczny szeregowy protokół. Magistrala jest "bezczynna", gdy obie linie są wysokie (otwarty kolektor). Jest to jedyny stan, w którym klawiatura / mysz mogą rozpocząć przesyłanie danych. Host ma najwyższą kontrolę nad magistralą i może zablokować komunikację w dowolnym momencie.

Urządzenie zawsze generuje sygnał zegara. Jeśli host chce przesłać dane, musi najpierw zablokować komunikację z urządzenia przez przestawienie zegara w stan niski. Host zwalnia zegar. Jest to stan "Request-to-Send" sygnalizujący urządzeniu rozpoczęcie generowania impulsów zegarowych.

Komunikacja Host-Urządzenia

1) Bring the Clock line low for at least 100 microseconds.
2) Bring the Data line low.
3) Release the Clock line.
4) Wait for the device to bring the Clock line low.
5) Set/reset the Data line to send the first data bit
6) Wait for the device to bring Clock high.
7) Wait for the device to bring Clock low.
8) Repeat steps 5-7 for the other seven data bits and the parity bit
9) Release the Data line.
10) Wait for the device to bring Data low.
11) Wait for the device to bring Clock low.
12) Wait for the device to release Data and Clock

Zapoznanie się z interfejsem łącza PS/2 dla obsługi myszki

Interfejs PS/2 dla myszki wykorzystuje dwukierunkowy protokół szeregowy do przesyłania danych o ruchu i danych przycisku do kontrolera urządzenia zewnętrznego komputera. Kontroler, w odpowiedzi, może wysłać kilka poleceń do myszy, aby ustawić szybkość raportu, rozdzielczość, zresetować mysz, wyłączyć mysz. Host dostarcza myszy zasilanie 5V. Myszka PS/2 używa tego samego protokołu co klawiatura PS/2.

Standardowy interfejs myszy PS / 2 obsługuje : ruch X (prawy / lewy), ruch Y (w górę / w dół), lewy przycisk, środkowy przycisk i prawy przycisk. Mysz okresowo odczytuje te wejścia i aktualizuje różne liczniki i flagi, aby odzwierciedlić ruch i stany przycisków. W naszym projekcie zajmowałyśmy się odczytywaniem ruchu X oraz ruchu Y.

Standardowa mysz ma dwa liczniki, które śledzą ruch: licznik ruchu X i licznik ruchu Y. Są to wartości uzupełniające dla 9-bitowych 2-ek, a każda z nich ma przypisaną flagę przepełnienia. Ich zawartość wraz ze stanem trzech przycisków myszy są wysyłane do hosta w postaci 3-bajtowego pakietu danych ruchu.

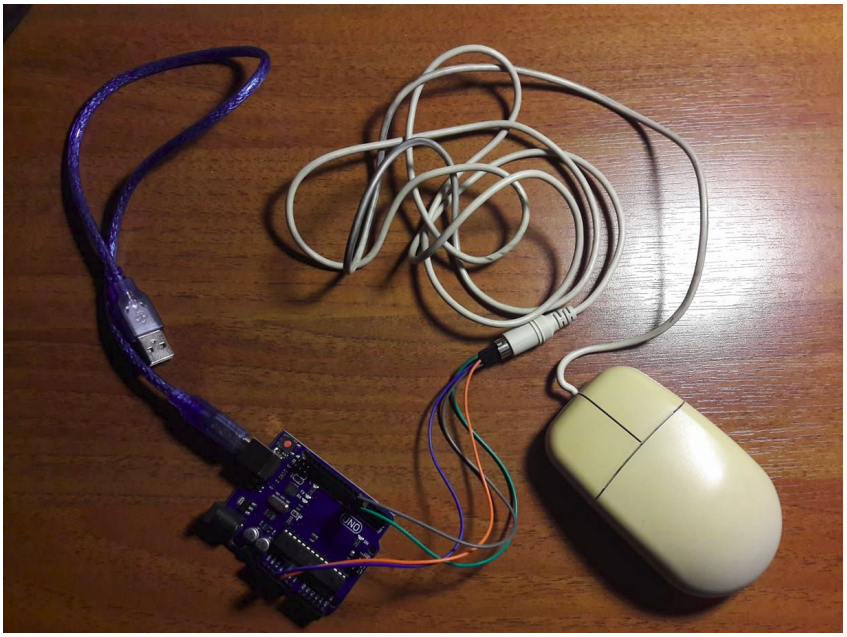
Kiedy mysz odczytuje swoje wejścia, zapisuje aktualny stan swoich przycisków i zwiększa / zmniejsza liczniki ruchu w zależności od ruchu, jaki nastąpił od ostatniego odczytu. Jeśli jeden z liczników przepełnił się, ustawiana jest odpowiednia flaga przepełnienia.

Movement Data Packet

Standardowa mysz PS / 2 przesyła informacje o ruchu / przycisku do hosta za pomocą następującego 3-bajtowego pakietu:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Y overflow	X overflow	Y sign bit	X sign bit	Always 1	Middle Btn	Right Btn	Left Btn
Byte 2	X movement							
Byte 3	Y movement							

Wartości ruchu są liczbami całkowitym, gdzie najbardziej znaczący bit pojawia się jako bit "znaku" w bajcie 1 pakietu danych ruchu. Ich wartość reprezentuje przesunięcie myszy względem jego pozycji, gdy poprzedni pakiet został wysłany, w jednostkach określonych przez bieżącą rozdzielczość. Zakres wartości, które można wyrazić, wynosi od -255 do +255. Jeśli ten zakres zostanie przekroczony, ustawiany jest odpowiedni bit przepełnienia.



d. Odczyt z myszki

Kolejnym etapem było uzyskanie danych z myszki. Jako pierwsze należało uaktywnić myszkę, poprzez wysłanie do niej sekwencji bitów.

Programowanie myszki

Kierując się opisaną wyżej obsługą interfejsu PS/2 napisałyśmy kod. Funkcje SetHigh oraz SetLow ustawiają w stan wysoki lub niski piny. Użyłyśmy do tego funkcji z biblioteki Arduino - digitalWrite() oraz pinMode().

Funkcja Write() zapisuje dane do myszki, robi to w sposób opisany wyżej jako komunikacja Host-Urządzenie.

Funkcja Read() odczytuje dane z myszki.

Funkcja InitializeMouse() wysyła sekwencję bitów pozwalającą na uruchomienie myszki.

Funkcja mousePosition() odczytuje pozycję myszki, pierwszy bajt to znacznik rozpoczęcia odczytywania, otrzymujemy potwierdzenie, następny bajt to pakiet właściwości, a później odczytujemy X oraz Y.

digitalWrite()

[Digital I/O]

Description

Write a `HIGH` or a `LOW` value to a digital pin.

If the pin has been configured as an `OUTPUT` with `pinMode()`, its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for `HIGH`, 0V (ground) for `LOW`.

If the pin is configured as an `INPUT`, `digitalWrite()` will enable (`HIGH`) or disable (`LOW`) the internal pullup on the input pin. It is recommended to set the `pinMode()` to `INPUT_PULLUP` to enable the internal pull-up resistor. See the digital pins tutorial for more information.

If you do not set the `pinMode()` to `OUTPUT`, and connect an LED to a pin, when calling `digitalWrite(HIGH)`, the LED may appear dim. Without explicitly setting `pinMode()`, `digitalWrite()` will have enabled the internal pull-up resistor, which acts like a large current-limiting resistor.

pinMode()

Description

Configures the specified pin to behave either as an input or an output. See the description of digital pins for details on the functionality of the pins.

As of Arduino 1.0.1, it is possible to enable the internal pullup resistors with the mode INPUT_PULLUP. Additionally, the INPUT mode explicitly disables the internal pullups.

Kod

```
char x,y,opcje;
int XX,YY;

#define CLOCK  6
#define DATA   5

void SetHigh(int pin){
  pinMode(pin, INPUT);
  digitalWrite(pin, HIGH);
}

void SetLow(int pin){
  pinMode(pin, OUTPUT);
  digitalWrite(pin, LOW);
}

void Write(unsigned char data){
  unsigned char parity=1;
  SetLow(CLOCK); //1
  delayMicroseconds(100); //1
  SetLow(DATA); //2
  SetHigh(CLOCK); //3
  while(digitalRead(CLOCK)==HIGH); //4
  for(int i=0; i<8; i++){
    if(data&0x01) SetHigh(DATA); //5
    else SetLow(DATA);
    while(digitalRead(CLOCK)==LOW); //6
    while(digitalRead(CLOCK)==HIGH); //7
    parity^=(data&0x01);
    data=data>>1;
  } //8
  if(parity) SetHigh(DATA); //9
  else SetLow(DATA);
  while(digitalRead(DATA)==HIGH); //10
  while(digitalRead(CLOCK)==HIGH); //11
  while((digitalRead(CLOCK)==LOW)&&(digitalRead(DATA)==LOW)); //12
}
```

```
unsigned char Read(void){
  unsigned char data=0, bit=1;
  SetHigh(CLOCK);
  SetHigh(DATA);
  delayMicroseconds(50);
  while(digitalRead(CLOCK)==HIGH);
  delayMicroseconds(5);
  while(digitalRead(CLOCK)==LOW);
  for(int i=0; i<8; i++){
    while(digitalRead(CLOCK)==HIGH);
    if(digitalRead(DATA)==HIGH) data|=bit;
    while(digitalRead(CLOCK)==LOW);
    bit=bit<<1;
  }
  while(digitalRead(CLOCK)==HIGH);
  while(digitalRead(CLOCK)==LOW);
  while(digitalRead(CLOCK)==HIGH);
  while(digitalRead(CLOCK)==LOW);
  SetLow(CLOCK);
  delayMicroseconds(100);
  return data;
}

void MousePosition(char &x, char &y){
  Write(0xEB); //Read Data
  Read(); // potwierdzenie FA
  //Read(); // 1Byte
  opcje = Read();
  x = Read();
  y = Read();
}

void InitializeMouse(void){
  Write(0xFF);
  for(int i=0; i<3; i++) Read();
  Write(0xF0); //a moze 0xFF
  Read();
  delayMicroseconds(100);
  XX = 0;
  YY = 0;
}
```

```
void setup() {
  SetHigh(CLOCK);
  SetHigh(DATA);
  Serial.begin(115200);
  while(!Serial);
  Serial.println("Do biegu");
  InitializeMouse();
  Serial.println("Gotowi!");
  Serial.println("Start!");
}

void loop() {
  int yInt = 0;
  int xInt = 0;
  MousePosition(x,y);
  //Serial.print("Opcje = ");
  //Serial.print(opcje, BIN);
  if(opcje&0x07 == 1) { //jezeli jest overflow w y
    yInt = int(y) + 256;
  }
  if(bitRead(opcje,6) == 1) { //jezeli jest overflow w x
    xInt = int(x) + 256;
  }

  XX = XX + x + xInt;
  YY = YY + y + yInt;

  Serial.print("\tX = ");
  Serial.print(x, DEC);
  Serial.print("\tXX = ");
  Serial.print(XX);
  Serial.print("\tY = ");
  Serial.print(y, DEC);
  Serial.print("\tYY = ");
  Serial.println(YY);

  delay(1000);
}
```

Podsumowanie i wnioski

Projekt pozwolił nam zapoznać się z programowaniem niskopoziomowym. Musiałyśmy dokładnie poznać działanie magistrali PS/2. Dzięki temu uświadomiłyśmy sobie, jak działa urządzenie, którym posługujemy się na codzień. Początkowo odczytywanie współrzędych miało być jedynie jedną z części naszego projektu, lecz nie udało nam się go w całości zrealizować. Pozwala nam to jednak na rozbudowę go w przyszłości.