Załącznik nr 2

Kod źródłowy pliku max6675.h

```
// this library is public domain. enjoy!
// www.ladyada.net/learn/sensors/thermocouple

#if ARDUINO >= 100
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif

class MAX6675 {
 public:
  MAX6675(int8_t SCLK, int8_t CS, int8_t MISO);

  double readCelsius(void);
  double readFahrenheit(void);
  // For compatibility with older versions:
  double readFarenheit(void) { return readFahrenheit(); }
 private:
  int8_t sclk, miso, cs;
  uint8_t spiread(void);
};
```

Kod źródłowy pliku max6675.cpp

```
// this library is public domain. enjoy!
// www.ladyada.net/learn/sensors/thermocouple

#ifdef __AVR
  #include <avr/pgmspace.h>
#elif defined(ESP8266)
  #include <pgmspace.h>
#endif
#include <util/delay.h>
#include <stdlib.h>
#include "max6675.h"

MAX6675::MAX6675(int8_t SCLK, int8_t CS, int8_t MISO) {
  sclk = SCLK;
  cs = CS;
  miso = MISO;

  //define pin modes
  pinMode(cs, OUTPUT);
  pinMode(sclk, OUTPUT);
  pinMode(miso, INPUT);

  digitalWrite(cs, HIGH);
}
double MAX6675::readCelsius(void) {

  uint16_t v;

  digitalWrite(cs, LOW);
  _delay_ms(1);

  v = spiread();
  v <<= 8;
```

```
  v |= spiread();

  digitalWrite(cs, HIGH);

  if (v & 0x4) {
    // uh oh, no thermocouple attached!
    return NAN;
    //return -100;
  }

  v >>= 3;

  return v*0.25;
}

double MAX6675::readFahrenheit(void) {
  return readCelsius() * 9.0/5.0 + 32;
}

byte MAX6675::spiread(void) {
  int i;
  byte d = 0;

  for (i=7; i>=0; i--)
  {
    digitalWrite(sclk, LOW);
    _delay_ms(1);
    if (digitalRead(miso)) {
      //set the bit to 0 no matter what
      d |= (1 << i);
    }

    digitalWrite(sclk, HIGH);
    _delay_ms(1);
  }

  return d;
}
```

Kod źródłowy pliku Button.h

```
/*
||
|| @file Button.h
|| @version 1.6
|| @author Alexander Brevig
|| @contact alexanderbrevig@gmail.com
||
|| @description
|| | Provide an easy way of making buttons
|| #
||
|| @license
|| | Copyright (c) 2009 Alexander Brevig. All rights reserved.
|| | This code is subject to AlphaLicence.txt
|| | alphabeta.alexanderbrevig.com/AlphaLicense.txt
|| #
||
*/


#include "Arduino.h"
#define PULLUP HIGH
#define PULLDOWN LOW

#define CURRENT 0
#define PREVIOUS 1
#define CHANGED 2

class Button{
  public:
    Button(uint8_t buttonPin, uint8_t buttonMode=PULLDOWN);
    void pullup();
    void pulldown();
    bool isPressed();
    bool wasPressed();
    bool stateChanged();
    bool uniquePress();
    unsigned long timePressed();
  private:
    uint8_t pin;
    uint8_t mode;
    uint8_t state;
};
```

Kod źródłowy pliku Button.cpp

```
//include the class definition
#include "Button.h"

// <<constructor>>
// @parameter buttonPin sets the pin that this switch is connected to
// @parameter buttonMode indicates PULLUP or PULLDOWN resistor

Button::Button(uint8_t buttonPin, uint8_t buttonMode){
  this->pin=buttonPin;
  pinMode(pin,INPUT);
  buttonMode==PULLDOWN ? pulldown() : pullup();
  state = 0;
  bitWrite(state,CURRENT,!mode);
}
```

```
// Set pin HIGH as default
void Button::pullup(void){
  mode=PULLUP;
  digitalWrite(pin,HIGH);
}


// Set pin LOW as default
void Button::pulldown(void){
  mode=PULLDOWN;
}


// Return the bitWrite(state,CURRENT, of the switch
bool Button::isPressed(void){
    bitWrite(state,PREVIOUS,bitRead(state,CURRENT));
    if (digitalRead(pin) == mode){
        bitWrite(state,CURRENT,false);
    } else {
        bitWrite(state,CURRENT,true);
    }
    if (bitRead(state,CURRENT) != bitRead(state,PREVIOUS)){
        bitWrite(state,CHANGED,true);
    }else{
        bitWrite(state,CHANGED,false);
    }
  return bitRead(state,CURRENT);
}

// Return true if the button has been pressed
bool Button::wasPressed(void){
    if (bitRead(state,CURRENT)){
        return true;
    } else {
        return false;
    }
}

// Return true if state has been changed
bool Button::stateChanged(void){
    return bitRead(state,CHANGED);
}

// Return true if the button is pressed, and was not pressed before
bool Button::uniquePress(void){
    return (isPressed() && stateChanged());
}


// Zwraca czas od początku naciśnięcia przycisku do jego zwolnienia w milisekundach
unsigned long Button::timePressed(void) {
  static unsigned long start = 0;
  static unsigned long stop = 0;
  if (uniquePress())
    start = millis();
  if (!isPressed()) start = millis();

  stop = millis();
  if (!isPressed()) return 0;
  else return (stop - start);
}
```