Raspberry PI i I²C

W oparciu o: Arduino, PCF8591 i MCP23008

Tomasz Kopacz

Microsoft

Scenariusz

Szyna I^2C pozwala dołączać w łańcuchu wiele urządzeń używając tylko 2 pinów GPIO

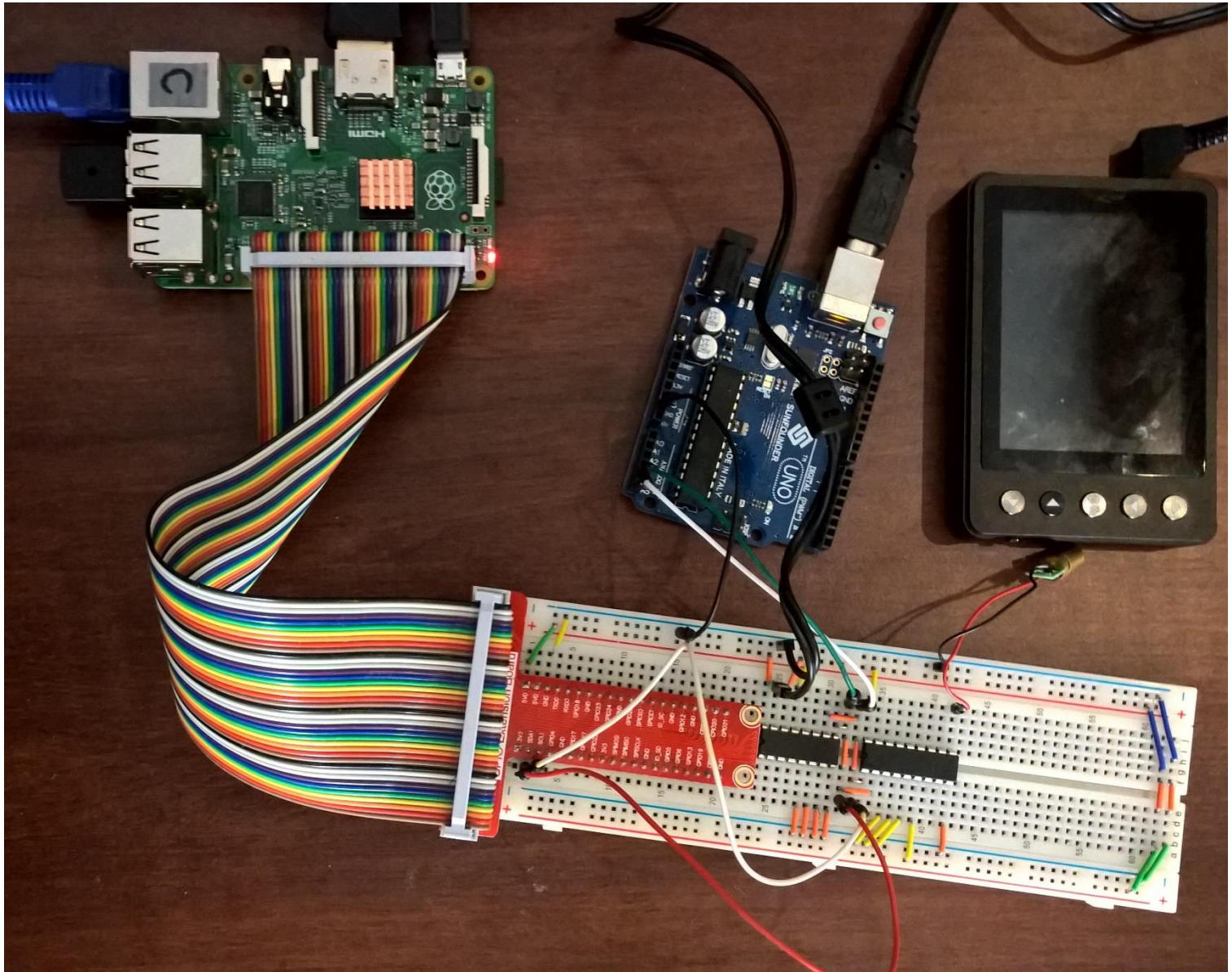
W tym scenariuszu połączymy:

PCF8591 – 8 bitowy konwerter A/D i D/A

MCP23008 – extender z dodatkowymi 8 portami GPIO

Arduino Uno (może być inne) – tu używane jako kalkulator, ale… otwiera świat na różne urządzenia które już są obsługiwane przez Arduino

Budowane urządzenie



Przypomnienie: Windows IoT - API (użytkowe)

GpioPin

- Bierze się z GpioController
- Input (potem GpioPin.Read) | Output (potem GpioPin.Write)
- RisingEdge | FallingEdge
- ValueChanged – reaguje na zmianę wartości

I2cDevice

- Bierze się z I2cConnectionSettings | I2cDevice.GetDeviceSelector | DeviceInformation.FindAllAsync | I2cDevice.FromIdAsync
- Read | ReadPartial
- Write | WritePartial
- WriteRead
- WriteReadPartial
- 2 piny (+ masa), StandardMode: 100 kbit/s, FastMode:1 mbit/s (są szybsze, ale to na razie to co jest dostępne)

SpiDevice

- Bierze się z SpiConnectionSettings | GetDeviceSelector | DeviceInformation.FindAllAsync | SpiDevice.FromIdAsync
- TransferFullDuplex
- TransferSequential
- Read | Write
- 4 piny (+ masa), prędkości większe niż I2C, do 100MHz, testy pokazują około 13 MB/s (~104 mbit/s)

Raspberry PI 2 - piny i ich znaczenie

3.3V PWR	1		2	5V PWR
I2C1 SDA	3		4	5V PWR
I2C1 SCL	5		6	GND
GPIO 4	7		8	Reserved
GND	9		10	Reserved
SPI1 CS0	11		12	GPIO 18
GPIO 27	13		14	GND
GPIO 22	15		16	GPIO 23
3.3V PWR	17		18	GPIO 24
SPI0 MOSI	19		20	GND
SPI0 MISO	21		22	GPIO 25
SPI0 SCLK	23		24	SPI0 CS0
GND	25		26	SPI0 CS1
Reserved	27		28	Reserved
GPIO 5	29		30	GND
GPIO 6	31		32	GPIO 12
GPIO 13	33		34	GND
SPI1 MISO	35		36	GPIO 16
GPIO 26	37		38	SPI1 MOSI
GND	39		40	SPI1 SCLK

GPIO

I²C: 3,5

SPI0: 19,21,23

ew: 24,25

(SPI1 zajęte)

PCF8591 – przetwornik A/D i D/A

6 PINNING

SYMBOL	PIN	DESCRIPTION
AINO	1	analog inputs (A/D converter)
AIN1	2	
AIN2	3	
AIN3	4	
A0	5	hardware address
A1	6	
A2	7	
V_{SS}	8	negative supply voltage
SDA	9	I²C-bus data input/output
SCL	10	I²C-bus clock input
OSC	11	oscillator input/output
EXT	12	external/internal switch for oscillator input
AGND	13	analog ground
V_{REF}	14	voltage reference input
AOUT	15	analog output (D/A converter)
V_{DD}	16	positive supply voltage

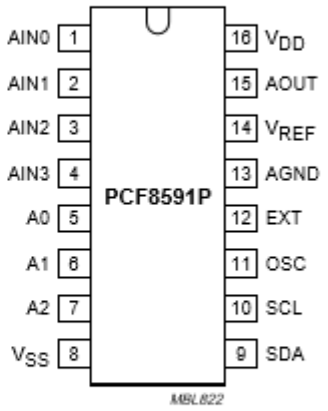


Fig.2 Pinning diagram (DIP16).

Datasheet tu

4 wejścia analogowe

1 wyjście analogowe

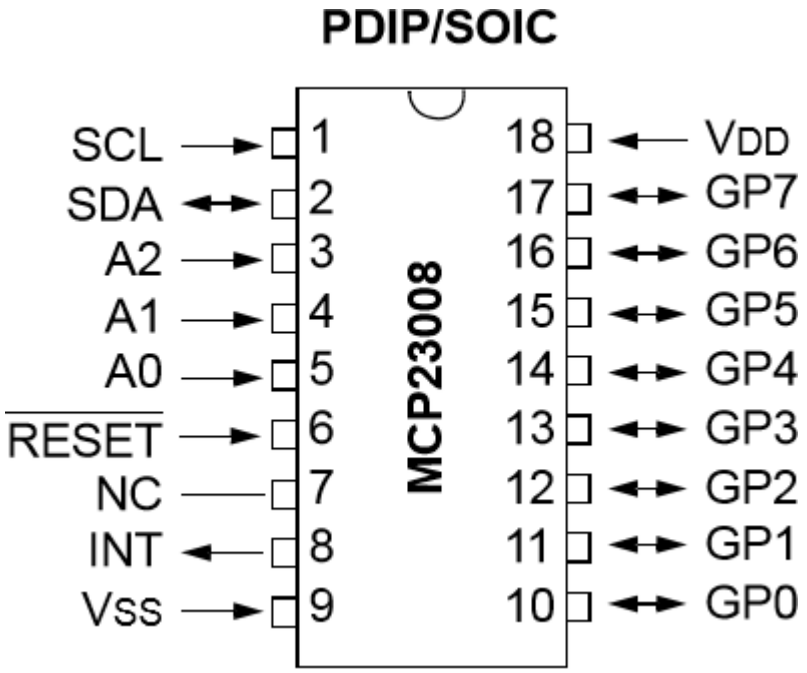
Adres 0x20

Gdy A0, A1, A2 dołączone do masy

Aby ustawić AOUT:

Wysłać do I2C: 0x40, <wartość>

MCP23008 – rozszerzenie GPIO



[Datasheet tu]

[Przykład Windows IoT]

8 wyjść i wejść GPIO

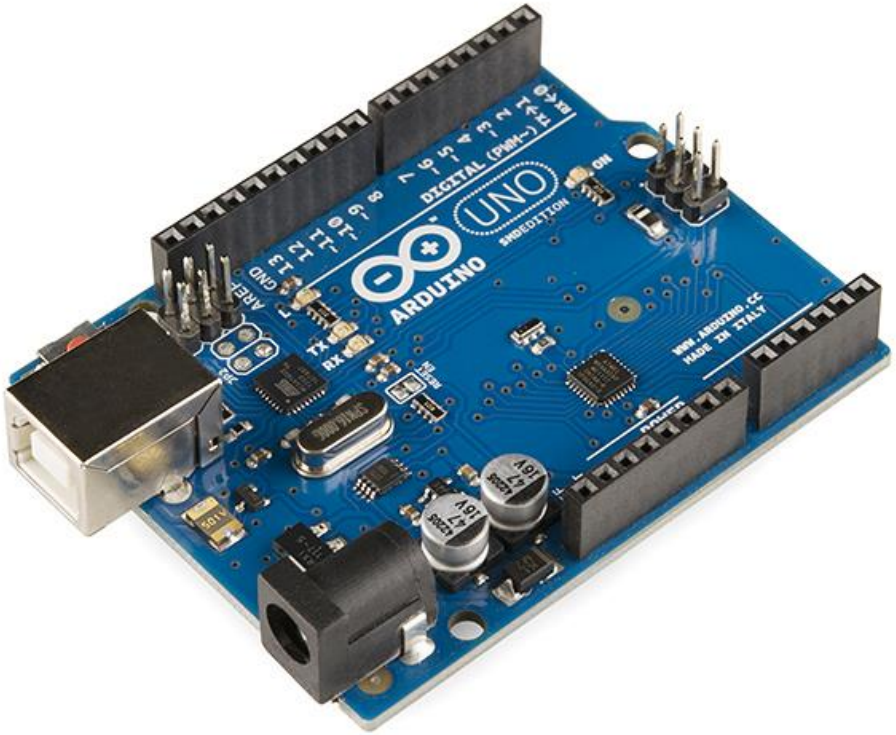
Adres 0x48

Gdy A0, A1, A2 dołączone do masy

Aby zmienić stan GP0:

Wysłać od I^2C: 0x0A, maska bitowa

Arduino



Używana biblioteka Wire

A4 – SDA

A5 – SCL

(masa)

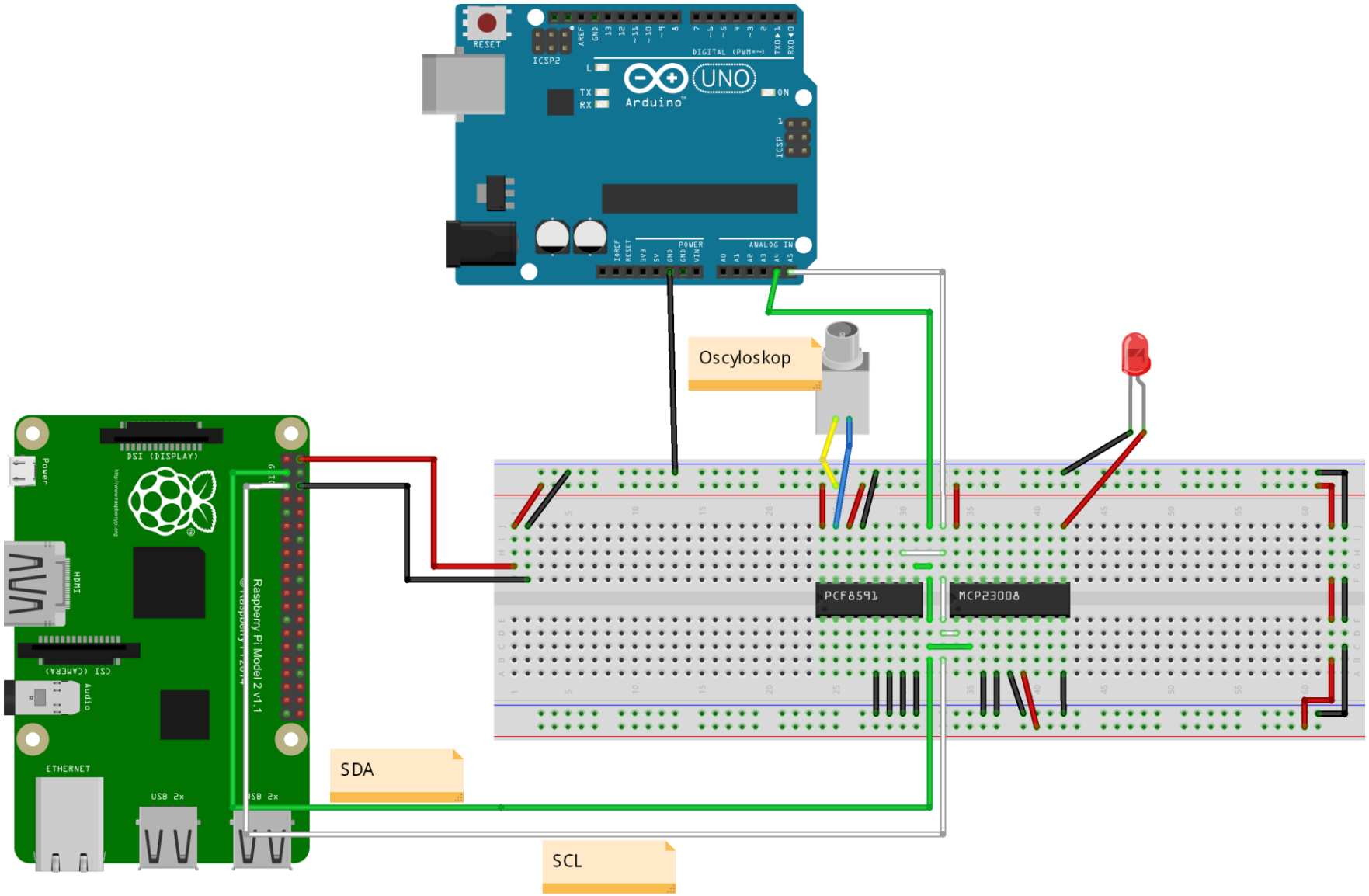
Adres 17 (wybrany)

```
#include <Wire.h>
Wire.onReceive(receiveEvent);
Wire.onRequest(requestEvent);
Wire.begin(17); //Jesteśmy slave numer 17

void receiveEvent(int howMany) {
  while (Wire.available() > 0) {
   byte b = Wire.read();
} }

void requestEvent() {  Wire.write(arr,2); }
```

Połączenia na diagramie



Połączenia z komentarzem

Protokół I²C

Dobry opis: https://learn.sparkfun.com/tutorials/i2c

Węzeł **Master** generuje cykle zegara synchronizujące komunikację. Inicjuje komunikację do **slave**

Węzeł **Slave** pracuje zgodnie z otrzymanymi cyklami zegara. Ma adres (liczba 7-bitowa). Gdy otrzyma komunikat (bajt) może zwrotnie wysłać odpowiedź. Ale **master** musi wydać polecenie „Read".

(W skrócie) sposób komunikacji:

Master wysyła do danego **slave** kod sterujący a **slave** wykonuje daną operacje opcjonalnie dodatkowo zwraca jakieś wartości do Master.

Czyli, jeżeli 2 urządzenia **slave** chcą coś do siebie wysłać, musi koordynować to **master**.

Rasperry PI 2, MinnowBoard MAX, .NET Micro Framework to urządzenia typu **master**

Są przykłady softwareowej realizacji I2C

Arduino może być zarówno **master** jak i **slave**

Warto rzucić okiem: https://www.arduino.cc/en/Tutorial/MasterWriter

Protokół I2C - z punktu widzenia Windows IoT:

Inicjalizacja:

```
string deviceSelector = I2cDevice.GetDeviceSelector();
var i2cDeviceControllers = await DeviceInformation.FindAllAsync(deviceSelector);
var i2cSettings = new I2cConnectionSettings(I2C_ADDR_MCP23008);
i2cMCP23008 = await I2cDevice.FromIdAsync(i2cDeviceControllers[0].Id, i2cSettings);
```

Wysłanie polecenia:

```
i2cMCP23008.Write(new byte[] { MCP23008_OLAT, olatRegister });
```

Wysłanie polecenia (kod `MCP23008_OLAT`) i odebranie wyniku do bufora `i2CReadBuffer`:

```
i2cMCP23008.WriteRead(new byte[] { MCP23008_OLAT }, i2CReadBuffer);
```

Odebranie wyniku, gdy możemy otrzymać niepełną informację:

```
var result = i2cArduino.ReadPartial(rbuffer);
```

Demo – działające rozwiązanie

Podsumowanie

I^2C pozwala podłączać wiele urządzeń

Używa tylko 2 pinów GPIO (+ masa)

I^2C jest używane:

Gdy ważniejsza prostota i cena niż prędkość.

Przykłady: odczyt danych konfiguracyjnych z SPD (pamięć), zarządzanie kartami PCI (SMBus 2.0), wyświetlacze OLED / LCD, czujniki diagnostyczne (temperatury), Display Data Channel w monitorach do ustawiania koloru, przetworniki analogowo/cyfrowe (te wolniejsze)

Dziękuję za wysłuchanie,

tkopacz@microsoft.com

Microsoft