

Raspberry PI i SPI

Tomasz Kopacz



Scenariusz

2 potencjometry (rezystor nastawny, regulowany dzielnik napięcia) podłączone do Raspberry PI

Zakres oporności – w zasadzie dowolny, np. 0-100K Ω , 0-50K Ω Obojętne czy skala liniowa czy logarytmiczna.

Raspberry PI odczytujące wskazania (analogowe) za pośrednictwem przetwornika analogowo-cyfrowego.

Tu używamy:

10-bitowego, 2 kanałowego **MCP 3002** podłączonego interfejsem SPI

Są też inne układy używające SPI:

MCP3008 ma 8 kanałów (wejść)

TLC1518 też ma 8 kanałów

LTC2452 ma 8 kanałów i rozdzielczość 16 bitową

MCP3202 ma 2 kanały i rozdzielczość 12 bitową.

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)

Przetwornik analogowo–cyfrowy (tu MCP3002)

Zasada działania: $\frac{V_{wejsciowe}}{V_{REF}} \times 1024$

$V_{wejsciowe}$ to mierzona wartość analogowa (zmieniana potencjometrem)

V_{REF} to referencyjne napięcie podane do układu

1024 – bo rozdzielczość MCP3002 to 10 bitów, $2^{10}=1024$

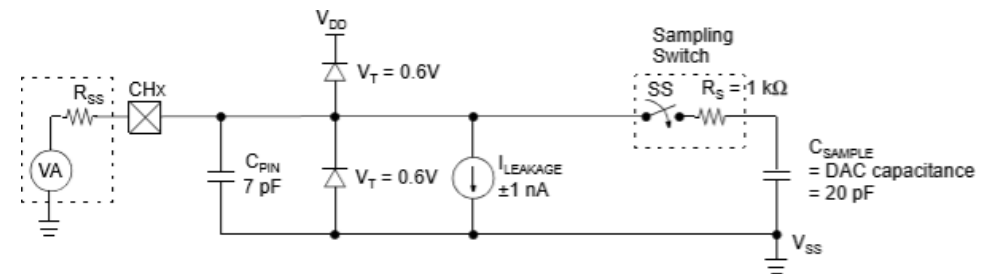
Uwaga!

W zależności od typu potencjometru i innych czynników – pomiary będą niedokładne.

Zwykle buduje się układ filtrujący by wyeliminować szum.

Na przykład układ z dokumentacji MCP3002:

W omawianym scenariuszu – pominięte.



Legend

VA	=	signal source
R _{SS}	=	source impedance
CHx	=	input channel pad
C _{PIN}	=	input pin capacitance
V _T	=	threshold voltage
I _{LEAKAGE}	=	leakage current at the pin due to various junctions
SS	=	sampling switch
R _S	=	sampling switch resistor
C _{SAMPLE}	=	sample/hold capacitance

MCP3002 – gdzie znaleźć dokumentacje

Ulubiona wyszukiwarka i np:

<http://www.alldatasheet.com/view.jsp?Searchword=Mcp3002>

All





Datasheet

Distributor

Manufacturer


MCP3002 Datasheet, PDF

Shortcut	MCP3002(22) recommended result.
Match, Like	MCP3002(3) MCP3002T(1)
Start with	MCP3002*(19) MCP3002~*(8) MCP3002T*(8) MCP3002_*(2)
End	No Data
Included	No Data
Manufacturer	

Electronic Manufacturer	Part no	Datasheet	Electronics Description
 Microchip Technology	MCP3002		MCP300X 10-Bit Analog-to-Digital Converters
	MCP3002		2.7V Dual Channel 10-Bit A/D Converter with SPI™ Serial Interface
	MCP3002		2.7V Dual Channel 10-Bit A/D Converter with SPI Serial Interface

MCP3002 Datasheet (PDF) - Microchip Technology



Part No.	MCP3002
Download	MCP3002 Click to view
File Size	439.99 Kbytes
Page	28 Pages
Maker	MICROCHIP [Microchip Technology]
Homepage	http://www.microchip.com
Logo	
Description	2.7V Dual Channel 10-Bit A/D Converter with SPI& Serial Interface

Jak podłączyć?

V_{DD} / V_{REF} do zasilania (tu 3.3V)

Układ może być zasilany 2.7 – 5.5V

CLK – do SPI0 CLK (zegar)

D_{OUT} do SPI0 MISO

D_{IN} do SPI0 MOSI

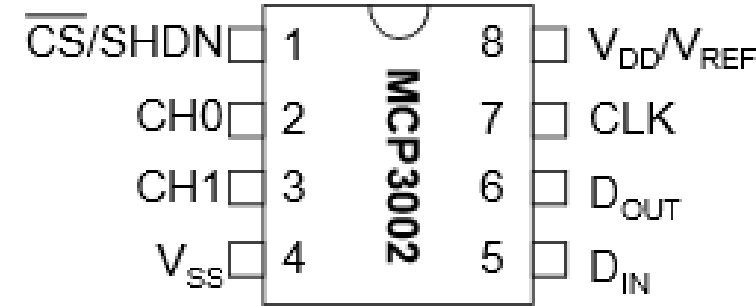
CS/SHDN do SPI0 CS0 (chip select, chip enable, CS , CE)

Możemy podłączyć jeszcze jeden układ – ale wtedy ta noga ma być połączona z SPI0 CS1. Pozostałe równolegle!

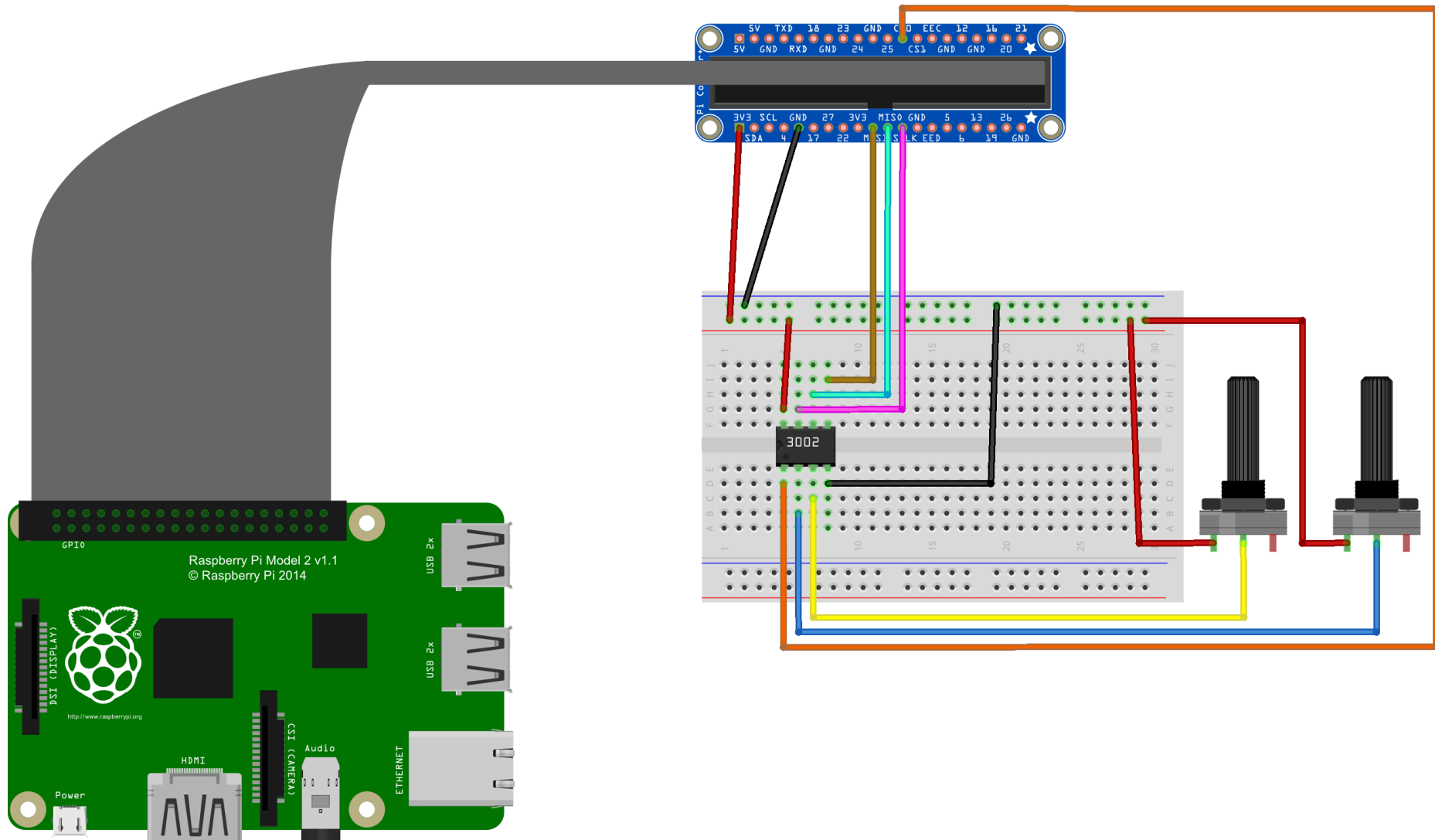
CH0 / CH1 – do źródeł analogowych (kanały)

Druga „noga” źródła analogowego do napięcia V_{REF} .

V_{SS} – do masy



Rysunek – może być łatwiej



Jak programować? I czytać...

Mode0 w SPI

Częstotliwość SPI: Max 3200000

Wysyłamy do SPI na przykład

0 1 10 1 000 0000 0000
0 1 11 0 000 0000 0000

puste start tryb kanał dopełnienie bajt_zero

Czyli: 0x68,0x00 lub 0x70,0x00

A potem odczytujemy
2 bajty w odpowiedzi i
znamy stan przetwornika

SPI zwraca tyle bajtów ile wysłaliśmy!

TABLE 5-1: Configuration Bits for the MCP3002.

	CONFIG BITS		CHANNEL SELECTION		GND
	SGL/DIFF	ODD/SIGN	0	1	
SINGLE ENDED MODE	1	0	+		-
	1	1		+	-
PSEUDO-DIFFERENTIAL MODE	0	0	IN+	IN-	
	0	1	IN-	IN+	

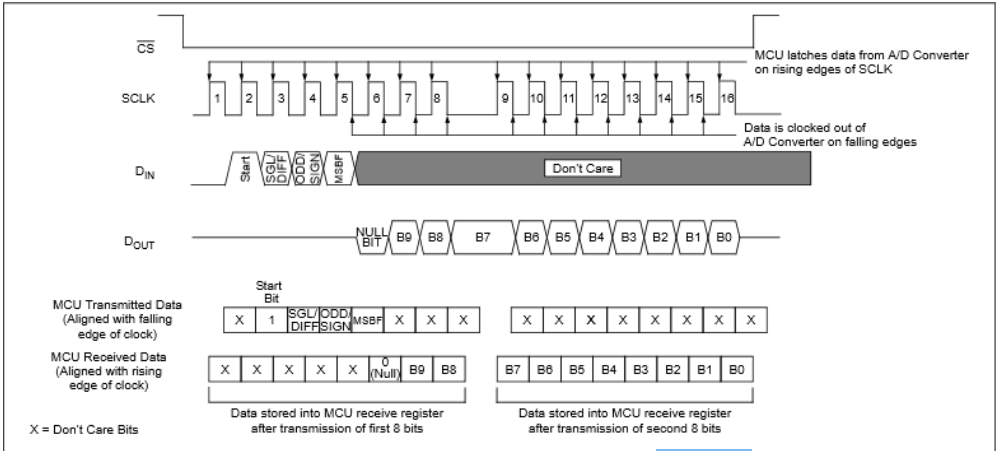


FIGURE 6-1: SPI Communication with the MCP3002 using 8-bit segments (Mode 0, 0: SCLK idles low).

(All Inputs/Outputs)						$T_{AMB} = 25^{\circ}C, f = 1\text{ MHz}$
Timing Parameters:						
Clock Frequency	f_{CLK}	—	—	3.2 1.2	MHz MHz	$V_{DD} = 5V$ (Note 2) $V_{DD} = 2.7V$ (Note 2)
Clock High Time	$t_{...}$	140	—	—	ns	

Inicjalizacja SPI

```
private const string SPI_CONTROLLER_NAME = "SPI0";
private const Int32 SPI_CHIP_SELECT_LINE = 0; /* SPI CS0, pin 24 */
SpiDevice m_spiDev;
...
var settings = new SpiConnectionSettings(SPI_CHIP_SELECT_LINE);
    settings.ClockFrequency = 3000000; // 3200000; 3000000
    settings.Mode = SpiMode.Mode0;

string spiAqs = SpiDevice.GetDeviceSelector(SPI_CONTROLLER_NAME);

var deviceInfo = await DeviceInformation.FindAllAsync(spiAqs);

m_spiDev = await SpiDevice.FromIdAsync(deviceInfo[0].Id, settings);
```

Odczyt

```
byte[] m_readBuffer = new byte[2];  
byte[] m_writeBufferCH0 = new byte[] { 0x68, 0x00}; //0 1 10 1 000  
...  
m_spiDev.TransferFullDuplex(m_writeBufferCH0, m_readBuffer);  
int resCH0 = convertToInt(m_readBuffer);
```

```
byte[] m_writeBufferCH1 = new byte[] { 0x70, 0x00}; //0 1 11 0 000  
...  
m_spiDev.TransferFullDuplex(m_writeBufferCH1, m_readBuffer);  
int resCH1 = convertToInt(m_readBuffer);
```

```
public int convertToInt(byte[] data) { //10 bitowe wejście, czyli 2 + 8 bitów  
    int result = data[0] & 0x03; result <<= 8; result += data[1];  
    return result; //0 - 1023  
}
```

Demo – działające rozwiązanie

Demo – przegląd kodu

Dygresja: Wiele urządzeń

Raspberry Pi 2 to tzw. SPI Master

Konieczny kanał Chip Select dla każdego urządzenia

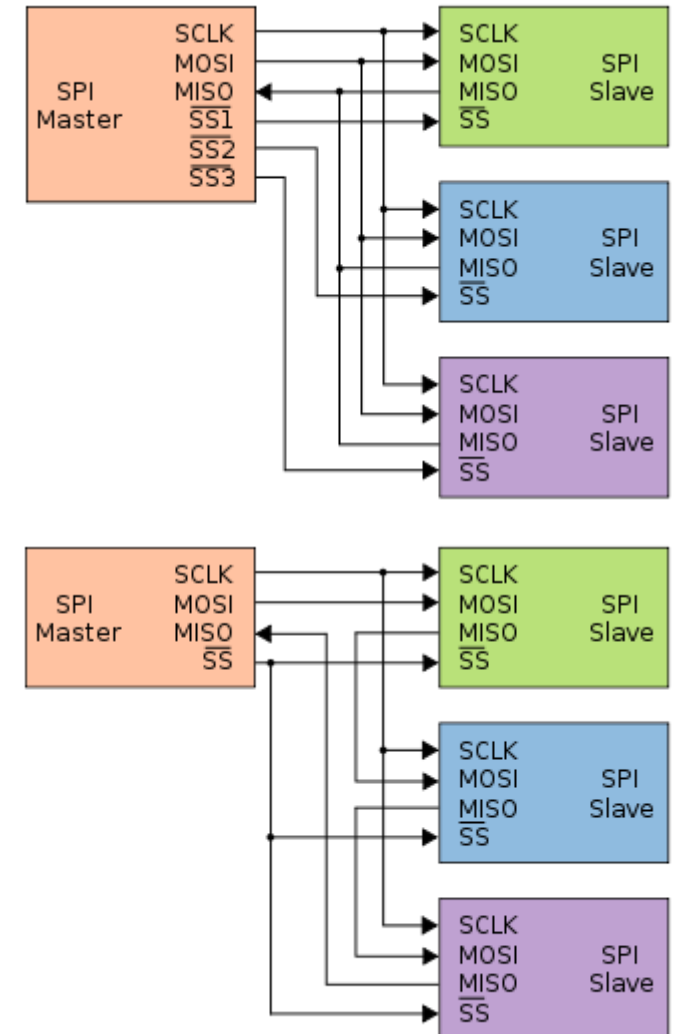
Chyba, że jak w dolnym rysunku, urządzenia ze sobą współpracują

Raspberry Pi 2 ma 2 kanały „Chip Select”

W naszym przykładzie stała SPI_CHIP_SELECT_LINE, wartość 0 lub 1 co odpowiada pinom 24 lub 26.

Inne interfejsy (np. I²C) mają znacznie wygodniejsze mechanizmy łączenia urządzeń w łańcuch

Ale są wolniejsze



Podsumowanie

Z punktu widzenia aplikacji w Windows 10 IoT po prostu wysyłamy ciąg bajtów do szyny SPI

SPI jest używane do komunikacji z:

Konwerterami ADC, Sensorami (temperatury, nacisku, wilgotności)

Wyświetlaczami (OLED, LCD)

Ekranami/panelami dotykowymi

Cyfrowymi potencjometrami

Pamięciami (zapis do EEPROM, BIOS itp.)

Kartami MMC / SD

...

Dziękuję za wysłuchanie,

tkopacz@microsoft.com

