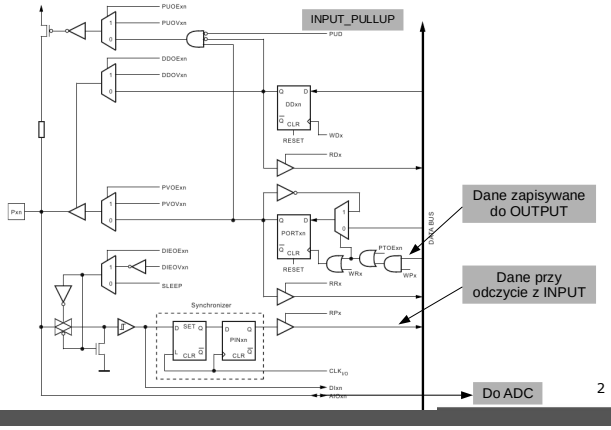
Inteligentne Systemy Pomiarowe

Wykład 2

mgr inż. Marek Wilkus
Wydział Inżynierii Metali i Informatyki Przemysłowej
AGH Kraków
http://home.agh.edu.pl/~mwilkus

Pin układu AVR

XxOV, xxOE - Override



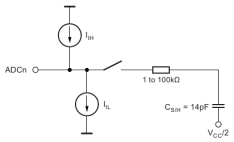
ADC

- Przetwornik analogowo-cyfrowy
- Dostępny jest w mikrokontrolerze (AVR, PIC) lub w postaci oddzielnego modułu.
- Umożliwia pomiar napięcia z reguły **względem masy**.
- Pomiary dokonywane są za pomocą porównania do **napięcia odniesienia**.

ADC

- Zmienny przebieg analogowy utrwalany jest na czas pomiaru za pomocą mechanizmu **sample and hold** – do pinu ADC elektronicznie dołączany jest kondensator (ok. 14pF) a na czas pomiaru jest on odłączany od mierzonego napięcia – mierzone jest napięcie na naładowanym kondensatorze.

Figure 23-8. Analog Input Circuitry

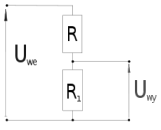


ADC w Arduino

- 10-bitowy ADC (przedział 0..1023).
- Multipleksowany na piny A0..A5.
- Domyślnie mierzy korzystając z napięcia zasilania jako napięcia odniesienia (co nie gwarantuje stabilności).
- Inne możliwości:
  - Z wewnętrznego, stabilizowanego 2.5V,
  - Z wewnętrznego, stabilizowanego 1.1V.
- Możliwość sprawdzenia własnego napięcia odniesienia dla 1.1V.

Pomiar większych napięć:

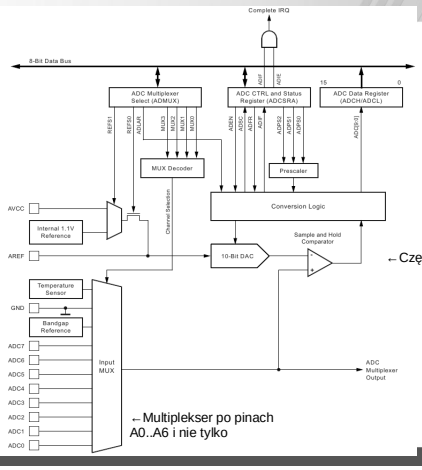
- Mamy zakres 0..5V. Do pmiaru większych napięć używamy **dzielnika napięcia**:
- Musimy pamiętać, by **masa była wspólna.**
- Rezystory nie muszą mieć wielkiej mocy, jednak powinny być dostosowane do napięcia.
- Pomiar wyłącznie DC (dla AC wymagane wyprostowanie ew. filtrowanie i odpowiednie przeliczenie wyników).



$$U_{wy} = \frac{U_{we}}{R + R_1} \cdot R_1$$

Źródło: Wikimedia Commons

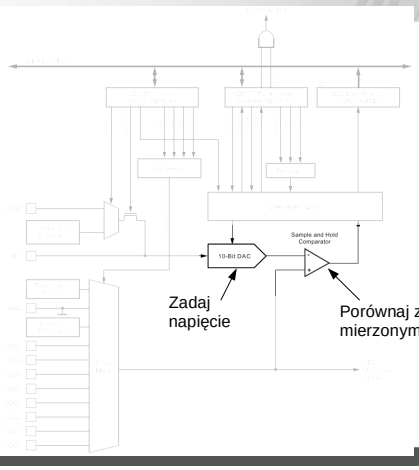
ADC w Arduino: Jak to działa?



←Część pomiarowa

←Multiplekser po pinach A0..A6 i nie tylko

ADC w Arduino: Jak to działa?



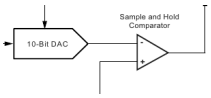
Tu odbywa się cały pomiar.

Zadaj napięcie

Porównaj z mierzonym

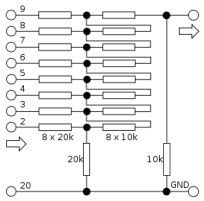
Jak ADC mierzy (i dlaczego tak długo?)

- Mierzona próbka przechowywana jest w kondensatorze sample-and-hold.
- DAC generuje kolejne napięcia.
- Komparator wychwytuje czy wygenerowane napięcie jest większe, czy mniejsze od mierzonego.
- Końcowy wynik transmitowany jest do pamięci.
- Generowanie kolejnych napięć zajmuje (w zależności od procesora) **12-18 cykli**.



Jak DAC generuje napięcie?

- PWM – z reguły nie jest stosowany – konieczność kalibracji.
- Dzielniki napięcia dla poszczególnych bitów
  - Stosowane już jako karty dźwiękowe (tzw. Covox / Disney Sound Source)
  - Łatwe do wbudowania w układ scalony,
  - Przy dobrze wykonanych rezystorach nie ma potrzeby kalibracji.



Korzystanie z ADC

- Co mierzyć? Względem czego? - ADMUX

Bit (0x7C)	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Shift results to the left.

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, internal V_{REF} turned off
0	1	AV_{CC} with external capacitor at
1	0	Reserved
1	1	Internal 1.1V voltage reference

MUX3..0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8[1]
1001	(reserved)
1010	(reserved)
1011	(reserved)
1100	(reserved)
1101	(reserved)
1110	1.1V (V_{BG})
1111	0V (GND)

Korzystanie z ADC

- Czy już mierzyć? - ADCSRA

Bit (0x7A)	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Włączyć ADC

Start pomiaru

Pomiar na wyzwalacz

Koniec pomiaru wyzwala przerwanie

Preskaler zegara systemowego dla ADC

Korzystanie z ADC

- Kiedy pomiar się skończył?
  - Bajt **ADCSRA**, bit „start konwersji” **ADSC** wraca na 0 po zakończeniu pomiaru.
- Wyniki dostępne są w **ADCL** i **ADCH**…
- ...ale najpierw należy odczytać ADCL! W przeciwnym wypadku dostaniemy w ADCL 0 lub wyniki z poprzedniego pomiaru.

Przykładowo: Pomiar temperatury

```
 ADMUX = _BV(REFS0) | _BV(REFS1)  | _BV(MUX3); // Mux na sensor temperatury
delay (20);  // Stabilizacja
ADCSRA |= _BV(ADSC);  // Start konwersji
while (bit_is_set(ADCSRA, ADSC));

byte low=ADCL;
int value=ADCH*256+low;

Serial.println (value - 350);
```

- Uwaga: Współczynnik odczytu do temperatury dobiera się doświadczalnie dla każdego egzemplarza układu osobno! Stąd jest bardzo zalecane korzystanie z pewniejszych urządzeń pomiaru temperatury.
  Ewentualnie dobierać dla każdego urządzenia np. na etapie sprawdzania po produkcji i zapisywać w EEPROMie, z którego odczytywany będzie na starcie.

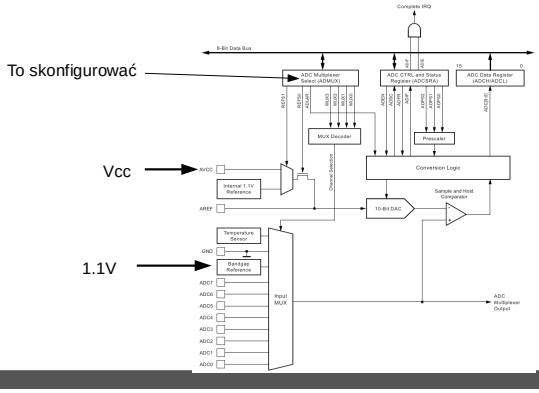
ADC mierzy własne zasilanie

- Domyślnie skonfigurowany ADC mierzy wszystko względem Vcc – jeżeli Vcc=5V, to 5V odczytywane jest jako 1023.
- Jeżeli Vcc wynosi 4.5V, to odczyt 4.5V to również 1023.
- Niezbędny jest inny punkt odniesienia…

- Możemy mierzyć napięcie wewnętrznego źródła 1.1V.
- Możemy jako punkt odniesienia wprowadzić pin AVCC podłączony do Vcc...

ADC mierzy własne zasilanie

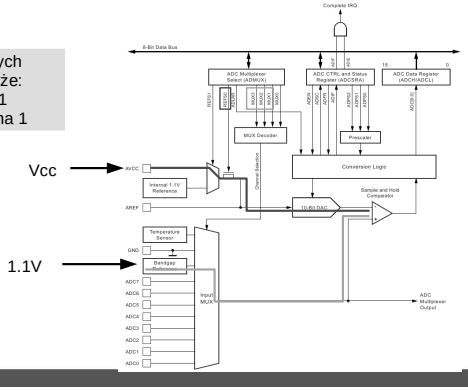
- Mierzymy 1.1V względem zasilania.



ADC mierzy własne zasilanie

- Mierzymy 1.1V względem zasilania.

Z zaznaczonych bitów wiemy, że:
- REFS0 na 1
- MUX1,2,3 na 1



Pomiar własnego zasilania

Z zaznaczonych bitów wiemy, że:
- REFS0 na 1
- MUX1,2,3 na 1

```
ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
delay(2); //Poczekaj na stabilizacje ADC

ADCSRA |= _BV(ADSC); // Start pomiaru

while (bit_is_set(ADCSRA,ADSC)); // czekaj
```

Musimy najpierw odczytać bajt LOW!

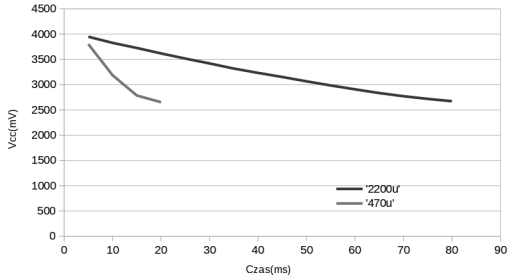
```
byte low=ADCL;
unsigned int result = ADCH*256 + low;
```

1.1V względem mierzonego napięcia !

```
result = 1.1*1023*1000 / result;
```

Ciekawostka

- Ile czasu jest w stanie działać Arduino korzystając z kondensatora?

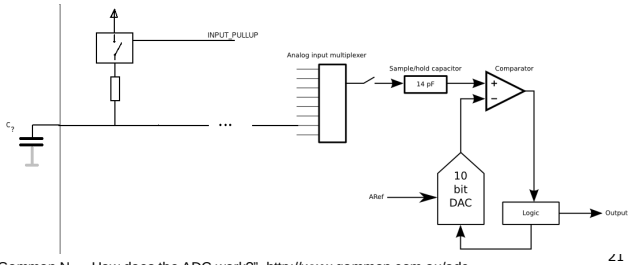


ADC i pomiar pojemności

- Po co w ogóle mierzyć pojemność?
  - Kalibracja obwodów wysokiej częstotliwości,
  - Samoczynna diagnoza systemów wysokiej niezawodności,
  - Sensory, ekrany i panele dotykowe,
  - Diagnostyka elementów.

Wykrywanie dotyku w AVR Przykładowa metoda

- Wykorzystujemy kondensator Sample-and-hold.
- Drugim kondensatorem jest pojemność dotkniętego (lub nie) elementu (np. fragmentu płytki pokrytego miedzią).
- Kondensatory ładujemy przez rezystor podciągający bo jest akurat poprawnie podłączony.



Źródło: Gammon N. - „How does the ADC work?" http://www.gammon.com.au/adc

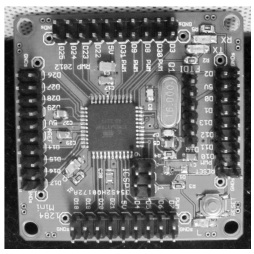
Implementacja

- Kalibracja: Według warunków zastosowania! (liczą się płytka, sąsiednie ścieżki, szerokość ścieżek, przelotki itd.).
- Przykładowy odczyt:

```
unsigned int readoff=0;
for (byte b=0;b<100;b++)
{
  pinMode(A0,INPUT_PULLUP);
  ADMUX = _BV(MUX3) | _BV(MUX2) | _BV(MUX1) | _BV(MUX0); //pomiar 0V
  ADCSRA |= _BV(ADSC);  // Start konwersji
  while (bit_is_set(ADCSRA, ADSC)); //kondensator sample and hold naładowany 5V.
  pinMode(A0,INPUT); //i po 5V przez INPUT_PULLUP...
  readoff=readoff+analogRead(A0); //konwencjonalny odczyt po uziemieniu przez C?
}
readoff=readoff/100;
Serial.println(readoff);
delay(500);
```


Zasilanie dla ADC

- Stabilne Vcc.
- Stabilne AREF, jeżeli używane.
- Porządne ścieżki na płytce drukowanej (nie prowadzić zasilania „włosowymi" ścieżkami do sygnałów).
- Kondensatory odsprzęgające. Na AREF również.
- Ewentualnie własny kondensator filtrujący.

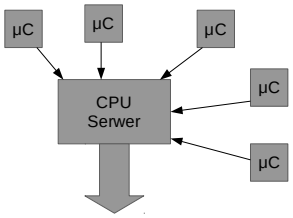


Komunikacja urządzeń

Więcej urządzeń pomiarowych

- Konieczność stosowania większej ilości urządzeń:
  - Wiele odległych punktów pomiarowych,
  - Rozproszony charakter stanowiska,
  - Trudności w przesyłaniu dłuższymi przewodami (tor analogowy!)
  - Możliwość zastosowania łącza bezprzewodowego.

Mikrokontroler jako „końcówka" pomiarowa



Podejmowanie działań,
Wyświetlanie,
Raportowanie...

+ Łatwość rozbudowy i naprawy,
+ Modularność,
+ Możliwość przejścia w architekturę magistrali,
+ Odporność na uszkodzenia (uszkadza się tylko jedna końcówka),
+ Możliwa nadmiarowość.

- Konieczność przemyślenia projektu systemu.
- Konieczność zaprojektowania protokołu.

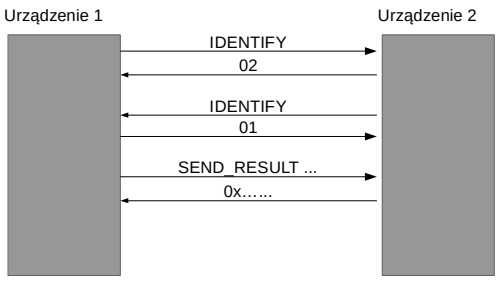
Standardy, standardy, standardy...

- RS232
  - Używany od dłuższego czasu, sprawdzony.
  - Tylko komunikacja punkt-punkt.
- RS485
  - Większa długość przewodów,
  - Możliwość pracy z wieloma urządzeniami naraz.
- CAN
  - Przystosowany do czujników szybko wymieniających dane.
  - Bardzo ściśle ustandaryzowany.
- I2C/SPI
  - Do zastosowań wewnątrz pojedynczego urządzenia.
- ARINC429
  - Wysoka niezawodność.
  - Kolejki priorytetów, możliwość nadawania przerwań.
  - Zastosowania: Lotnictwo, urządzenia medyczne i wojskowe

RS232 – jak obejść punkt-punkt?

- Linia BUSY
  - Żadne urządzenie nie może komunikować się, gdy magistrala jest zajęta.
  - Każde urządzenie może zając magistralę.
- Multipleksowanie urządzeń i buforowanie.
  - Jednostka centralna decyduje o „podłączeniu" urządzenia na port.
- Komunikacja łańcucha urządzeń.
  - Każde urządzenie ma 2 porty. Przekazuje wszelkie komunikaty do sąsiednich.
  - Możliwość rozszerzenia o więcej portów (sieć typu mesh).

Wymiana informacji



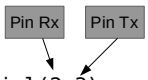
- Wymagane są minimum dwa elementy protokołu:
  - „Przedstawienie" się urządzenia (IDENTIFY).
  - Pozyskanie danych z urządzenia (SEND_RESULT).

Rozszerzenia

- Urządzenie może wysyłać również numer urządzenia docelowego. Dzięki temu możliwa jest praca na magistrali.
- Protokół może być opracowany w dowolnej postaci (zarówno znaki ASCII łatwe do debugowania jak i bajty charakterystyczne).
- Jedynie wywołania i odopwiedzi muszą być standardowe.

Arduino i drugi RS232

- Software'owy RS232: Biblioteka SoftwareSerial.
  - Brak buforowania.
- Użycie:




```
SoftwareSerial mySerial(2,3);

//w Setup:

mySerial.begin(4800);
```

A dalej jak z typowym portem szeregowym.

Przykładowa implementacja protokołu

```
SoftwareSerial mySerial(10, 11);

void setup()
{
  Serial.begin(9600);   //inicjalizacja sprzętowego
  mySerial.begin(4480); //inicjalizacja programowego
}
```

```
void loop()
{
  if (mySerial.available())  //Parsowanie poleceń z zewnątrz
  {
   byte k=mySerial.read();
   if (k=='I')               //IDENTIFY
   {
    mySerial.print('2');
   }
   if (k=='R')               //READ
   {
    mySerial.print(temp);
   }
  }
//-------------------------------------------------------------------------------
 if (Serial.available())    //Parsowanie polecenia od użytkownika
 {
  if (Serial.read()=='G')   //Polecenie: Odczytaj wszystkie wyniki
  {
    Serial.print(temp);                 //Pokaz temperature
    Serial.print(" but on device "); //ale na urządzeinu numer...

    mySerial.print('I');               //na szybko odczytaj numer podłączonej płytki
    int q=0;
    while ((!mySerial.available())&&(q<4000)) {q++;}; //dajemy czas na budowanie odpowiedzi
    Serial.print((char)mySerial.read());     //...Drukujemy pozyskany numer

    Serial.print(" it is ");

    mySerial.print('R');          //odczytaj temperaturę z zewnętrznego urządzenia
    q=0;
    while ((!mySerial.available())&&(q<4000)) {q++;}; //dajemy czas na budowanie odpowiedzi
    while (mySerial.available())
    {
     Serial.print((char)mySerial.read()); //Drukujemy kolejne znaki
     delay(3);                            //odstęp czasowy pomiędzy znakami
    }
    Serial.println();                   //koniec odpowiedzi dla użytkownika
  }
 }
}
```

Software Serial – częste pytania

- Użytkowanie do celów debuggowania?
  - Nie bez modyfikacji – W Arduino Uno połączenie USB jest podłączone na stałe do pinów 0 i 1.
- Łączenie dwóch płytek: Jak?
  - Połączyć masy
  - Rx → Tx
  - Tx → Rx
- Maksymalna prędkość?
  - Zależna od „obciążenia" procesora, bezpiecznie jest do ok. 19200bps.