WYŻSZA SZKOŁA BIZNESU
NATIONAL-LOUIS UNIVERSITY

Złożenie pracy online:
2009-05-20 13:37:00
Kod pracy:
1387
Kod załącznika:
1375

WYDZIAŁ INFORMATYKI
KIERUNEK: INFORMATYKA

Adrian Zarzeka
(Nr albumu: 9725*INF/LIC)

Serwer meteorologiczny - aplikacja wykorzystująca mikrokontrolery Atmel

Meteo Server - an application which uses Atmel microcontrolers

Praca licencjacka

Promotor: **dr Krzysztof Przybycień**

Nowy Sącz, 2009

Wstęp

Człowiek od zawsze fascynował się bardziej przyszłością niż przeszłością. Pragnieniem naszym było i jest odgadnięcie przyszłości w różnych jej wymiarach - zdarzeń ważkich i całkiem błahych. W przypadku prognozowania pogody aspekt praktyczny tego typu działań nie ulega wątpliwości. Ujawnił się on już z wielką siła w starożytności, gdy pierwsi filozofowie próbowali przewidzieć jaka będzie pogoda gdyż mogło to decydować o sprawach gospodarczych (handel, żegluga) czy nawet losach rozstrzygnięć wojennych. Wtedy też ujawnił się problem znany do dziś - pogoda jest trudno przewidywalna.

Celem tej pracy jest przedstawienie prostego mechanizmu działania stacji meteorologicznej – serwera, który wykorzystuje programowalne układy mikroprocesorowe firmy ATMEL użyte do wykonywania oraz przeliczania danych zebranych poprzez odpowiednie czujniki. Jako przykład – dopełnienie wykorzystana została prognoza Interdyscyplinarnego Centrum Modelowania Matematycznego i Komputerowego.

W rozdziale pierwszym została przedstawiona budowa układów pomiarowych oraz mikroprocesor i opis jego funkcji.

Rozdział 2 poświecony został analizie serwera , jego pracy oraz aplikacji w nim funkcjonujących.

Rozdział 3 Podsumowanie całej pracy oraz wyciągniecie wniosków dotyczących ewentualnej rozbudowy

ROZDZIAŁ 1 – budowa systemu i wybór technologii

W mojej pracy zastosowałem 4 moduły pomiarowe. Są to :
- moduł pomiaru ciśnieniomierza – cyfrowo-analogowy
- moduł pomiaru wilgotnościomierza – cyfrowo-analogowy
- moduł pomiaru nasłonecznienia – cyfrowo-analogowy
- moduł pomiaru temperatury - cyfrowy

1. Moduł ciśnieniomierza oraz wilgotnościomierza

Jest to układ oparty o klasyczny czujnik firmy Freescale Semiconductor's o symbolu MPXA 4115 ogólno dostępny na rynku jak i również rezystancyjny czujnik do pomiaru wilgotności o symbolu SYH2T.

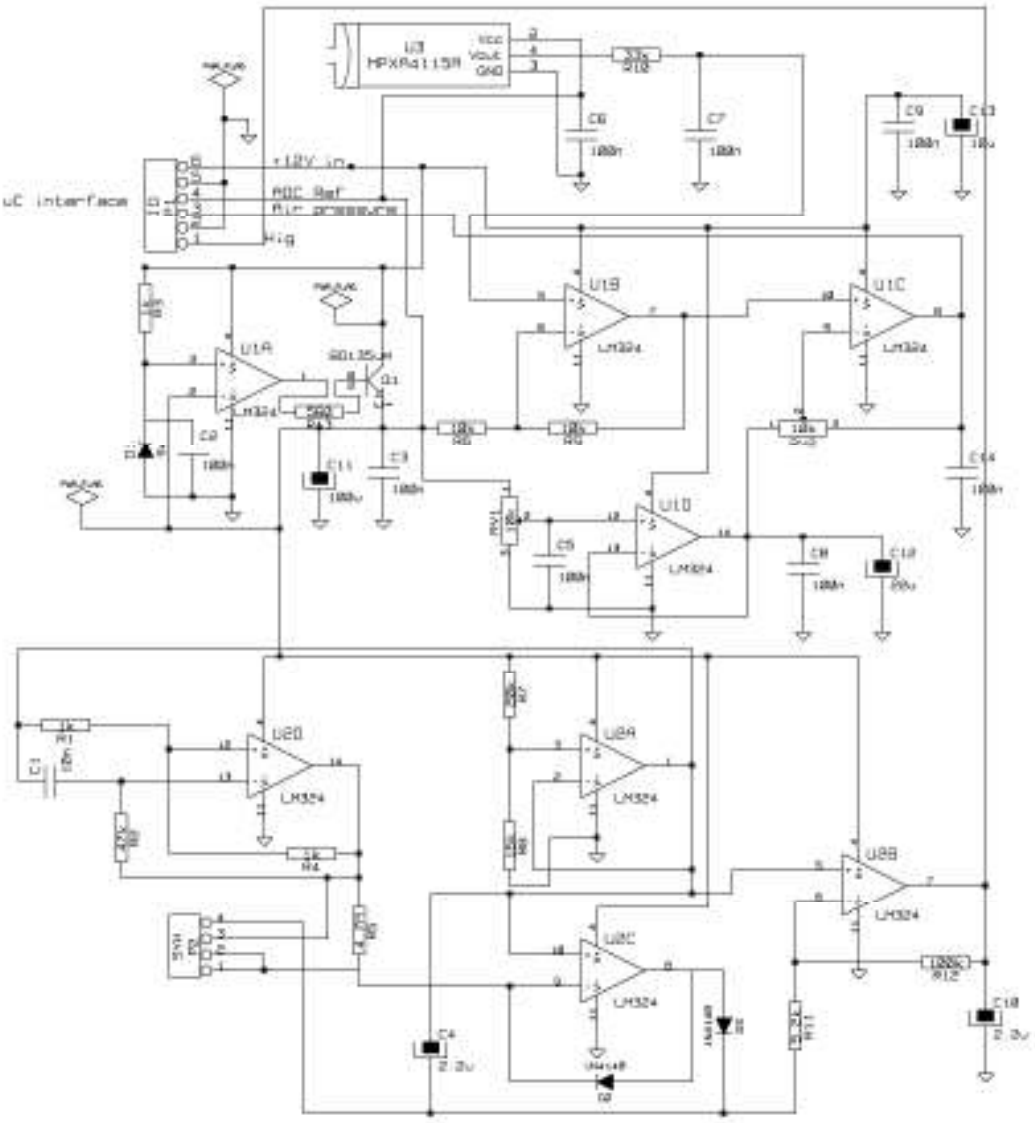
Parametry:
- zakres pomiarowy czujnika : *15-150 kilopaskali*
- zakres pomiarowy układu : *950-1050 hektopaskali (95-105kPa)*
- czułość układu: *45.9 miliwolt / 1 hektopaskal*
- maksymalny błąd pomiarowy *: 1,5%*
- temperatura pracy : *od -40 do +125 stopni Celsjusza*

Rysunek 1 Wygląd czujnika MPXA4115

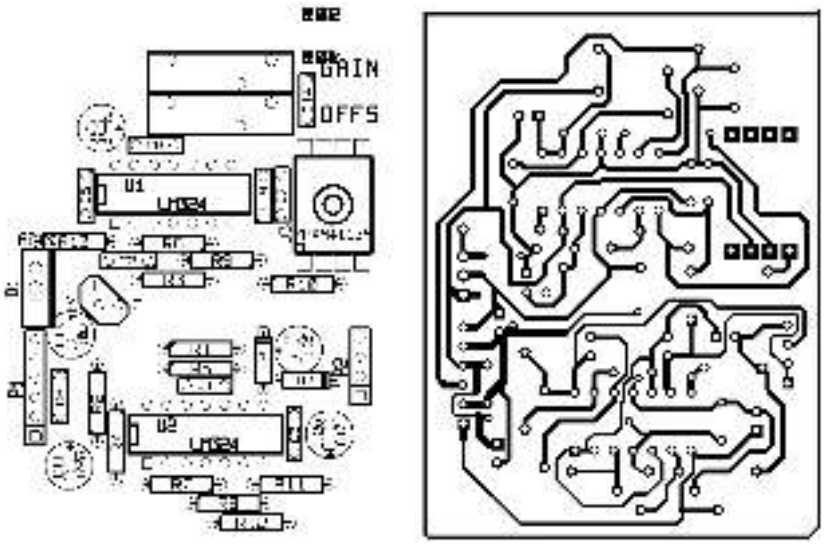
Rysunek 2 Czujnik wilgotności SYH-2T.

Obydwa te czujniki pracują w jednym module pomiarowym wyposażonym w wzmacniacze operacyjne aby pomiary były wiarygodne.

Schemat oraz projekt płytki modułu :



Rysunek 3. Schemat modułu wilgotnościomierza oraz ciśnieniomierza.



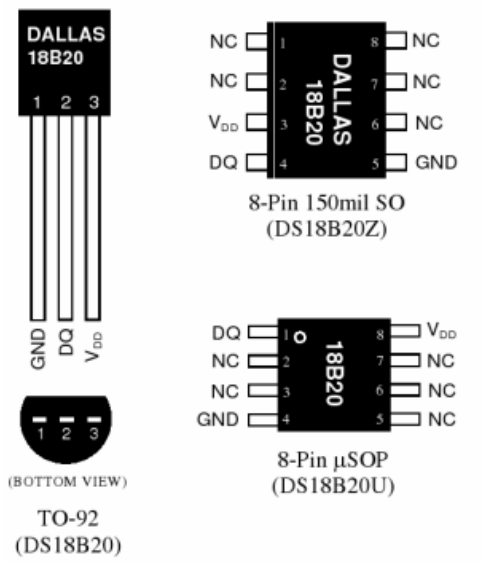
Rysunek 4. Widok płytki modułu wilgotności oraz ciśnienia

2. Moduł pomiaru temperatury

Jest to dobrze znany układ firmy Dallas o symbolu DS18B20

Parametry:

- Magistrala **1-Wire**
- Każdy z układów posiada własny **unikalny 128-bitowy kod**
- Może być zasilany napięciem **od 3V do 5.5V**
- Mierzy temperaturę **od -55 do +125 stopni**
- Zapewnia **dokładność 0.5 stopnia** dla temperatur -10 do 85 stopni
- **ustawiana rozdzielczość pomiaru 9 do 12 bitów**
- **odpowiedź czujnika - nie dłużej niż 750ms**



Rysunek 5. Czujnik DS18B20

Czujniki temperatury firmy Dallas cieszą się duża popularnością ze względu na ich prostotę użytkowania. W czujniku jak i w wielu układach Dallas'a wykorzystano technologie 1Wire (czyt. One Wire) gdzie komunikacja czujnika z jednostka obliczeniowa odbywa się w obu kierunkach tylko poprzez jeden przewód (oprócz masy) oznaczony jako „DQ". Każdy z podzespołów posiada własny unikalny 128 bitowy kod który służy do rozpoznawania urządzenia jeśli np. posiadamy magistrale na której pracuje 10 lub więcej czujników, układów łatwo w ten sposób dotrzeć do wybranego elementu.
Pomimo swojej prostoty, czujniki te sprawiają wiele problemów użytkownikom ze względu na rożne implementacje procedur obsługi 1Wire. Jedyny język programowania który posiada wbudowana implementacje to BASCOM.

Bascom to środowisko programistyczne przeznaczone do programowania mikrokontrolerów z serii 8051 i Atmel AVR w języku programowania o tej samej nazwie. Język ten jest zbliżony do języka Basic.

Kolejnym problemem który spotyka użytkowników to zawężanie zakresu pomiarowego tylko do temperatur dodatnich oraz odrzucania części ułamkowej. Związane jest to z niechęcią do zapoznania się z notą katalogową jak i również słabą znajomością języka angielskiego.

Zapoznajmy się więc z procedurami.

Wszystkie komendy podawane są w systemie szesnastkowym.

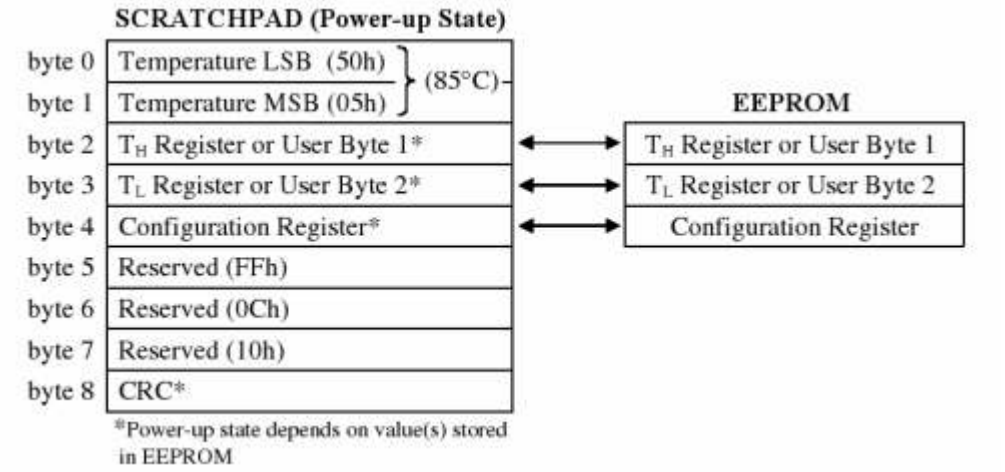
DS18B20 jest w stanie uśpienia, aby go wybudzić wysyłamy polecenie 0x44

Jest to rozkaz konwersji temperatury. Po pomiarze wynik jest zapisywany do dwóch bajtów rejestru (tzw. Scratchpad memory). Układ ponownie wchodzi w stan uśpienia.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LS Byte	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	bit 15	**bit 14**	**bit 13**	**bit 12**	**bit 11**	**bit 10**	**bit 9**	**bit 8**
MS Byte	S	S	S	S	S	2^6	2^5	2^4

Rysunek 6 Pierwsze 2 bajty pamięci

Pięć najstarszych bitów bajtu MSB na rysunku oznaczone literką „S” zawierają informacje o znaku temperatury natomiast trzy młodsze bity MSB oraz całą reszta bajtu LSB to 12 bitowa liczba wskazująca nam temperaturę po przemnożeniu jej przez 16. Jeżeli jest to liczba ujemna (temperatura poniżej zera) to wyrażona jest w kodzie binarnym U2 (uzupełnienie do dwóch).



Rysunek 7 Rejestr czujnika DS18B20

Komunikacja i rozkazy

Są podstawowe 3 kroki. Pierwszy to inicjalizacja czujnika poprzez zresetowanie magistrali 1WIRE. Następną rzeczą są komendy za pomocą których sczytujemy cała magistralę otrzymując w ten sposób numery seryjne poszczególnych czujników i oczywiście wywołanie konkretnie nas interesującego czujnika.

Do najbardziej znanych i używanych komend zaliczamy :
- Read ROM (polecenie 0x33) – czytanie 8 bitowego numeru seryjnego czujnika. Nie działa jeśli na magistrali jest ich więcej.
- Match ROM (polecenie 0x55) – umożliwia aktywowanie określonego układu na magistrali
- Skip ROM (polecenie 0xCC) – komenda pomijająca sprawdzanie numerów układów. Następne polecenia wysyłane będą do wszystkich czujników.

Ostatnim krokiem komunikacji jest wydanie konkretnego polecenia. Rozkazy te służą do konwersji temperatur oraz do zapisu i odczytu z rejestrów DS18B20.

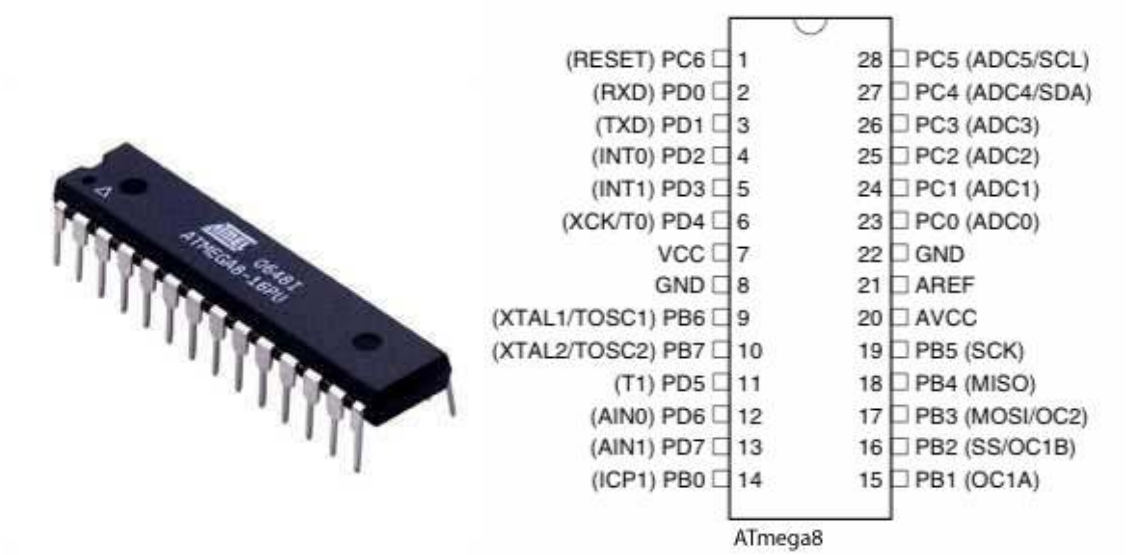
Do najważniejszych poleceń zaliczamy :
- Convert T (polecenie 0x44) – rozpoczyna proces konwersji temperatury, po jego zakończeniu wynik zostaje umieszczony w dwóch bajtach rejestru.

- Read Scratchpad (polecenie 0xBE) – powoduje rozpoczęcie odczytu danych począwszy od bitu 0 do bitu 8. Odczyt może być przerwany poprzez reset magistrali 1Wire jeżeli nie potrzebujemy wszystkich 9 bajtów.

3. Moduł pomiaru nasłonecznienia

Jest to ogniwo słoneczne wytwarzające napięcie poprzez promienie słoneczne. Zakres napięć ogniwa to 0 – 2,3V

4. Mikroprocesor



Rysunek 8 Serce układu – mikrokontroler ATmega8

Jest to mikroprocesor o wysokowydajnej architekturze AVR. Posiada on 8kb pamięci programowalnej flash w systemie ISP o trwałości do 10.000 cykli. Główny rdzeń CPU zapewnia nam poprawne i szybkie wykonywanie kodu a dla uzyskania pełnej wydajności, zbudowany został w oparciu o architekturę harwardzką (rozdzielone pamięci i szyny dla programu i danych). Wszystkie instrukcje wykonywane są potokowo tzn. podczas gdy jedna z instrukcji jest wykonywana, następna jest już pobierana z pamięci programu. Dzięki takiej pracy mikroprocesora możemy wykonać całą instrukcję w jednym takcie zegara.

Porty wejścia/wyjścia możemy konfigurować wedle naszych potrzeb. W projekcie procesor jest taktowany z częstotliwością 11.059Mhz co pozwala na szybką prace oraz dostosowanie do prędkości UART w celu uniknięcia błędów podczas transmisji.

Oto porty wykorzystane w tej pracy :

Pin portu	Nazwa systemowa pinu
PC 0	ADC0 (wejście ADC, kanał 0)
PC 1	ADC1 (wejście ADC, kanał 1)
PC 2	ADC2 (wejście ADC, kanał 2)
PC 3	ADC3 (wejście ADC, kanał 3)

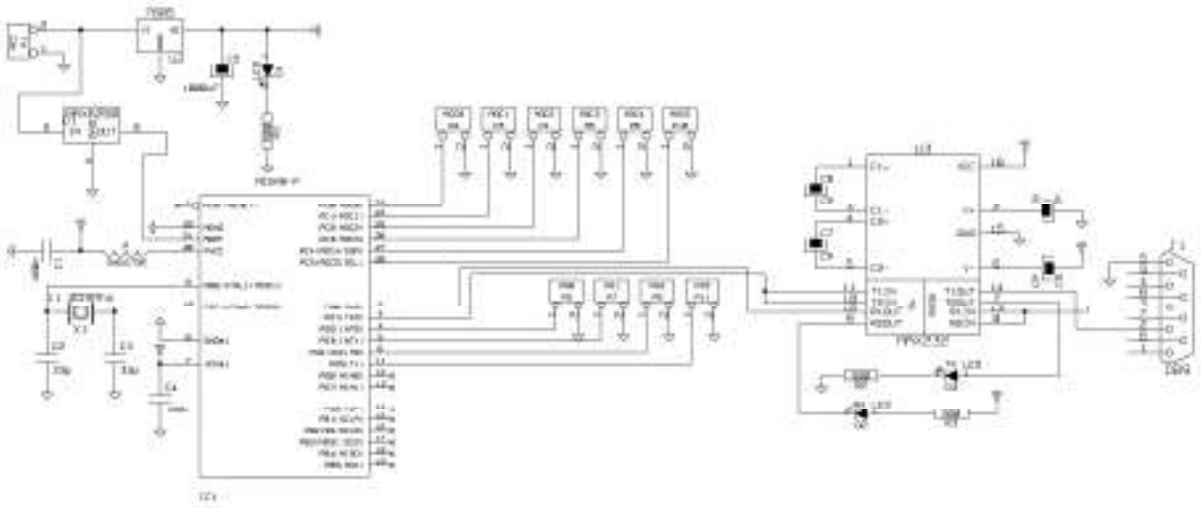
Tabela 1. Definicje portów analogowo-cyfrowych

Pin portu	Nazwa systemowa pinu
Pin 22	Masa zasilania
Pin 21	AREF (wejście napięcia odniesienia)
Pin 20	AVCC (zasilanie bloku ADC)
Pin 8	Masa zasilania
Pin 7	Zasilanie +5V

Tabela 2. Definicje portów bazowych

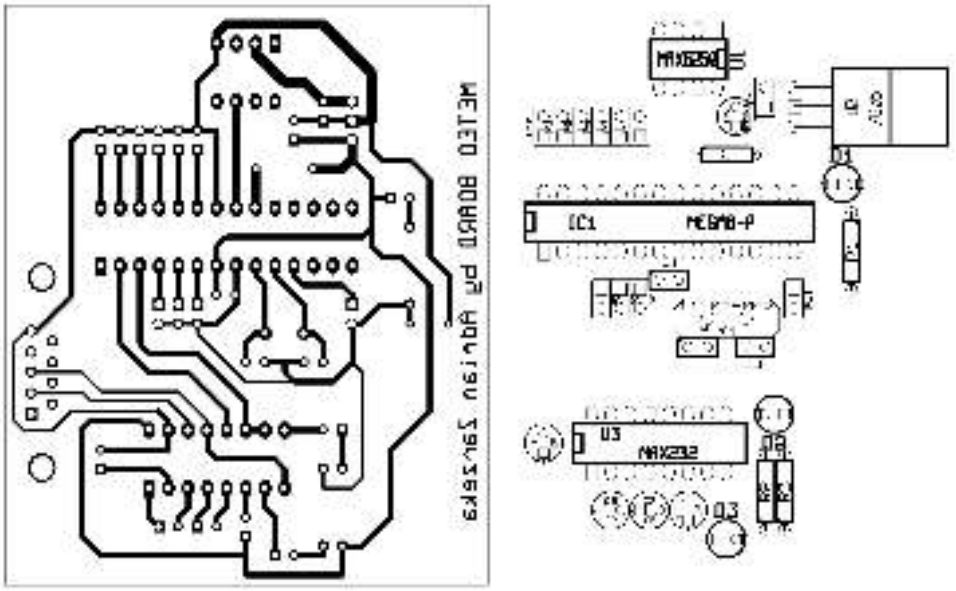
Pin portu	Nazwa systemowa pinu
PD 0	Uniwersalny port wejścia wyjścia
PD 1	Uniwersalny port wejścia wyjścia
PD 2	Uniwersalny port wejścia wyjścia
PD 3	Uniwersalny port wejścia wyjścia

Tabela 3. Definicje portów wejścia-wyjścia



Rysunek 9. Schemat płytki mikroprocesora

Jak widzimy cały układ jest dość prosty, składa się on tylko z układu ATmega8. Ponieważ poziomy sygnałów napięć po stronie mikrokontrolera różnią się, zastosowany został dodatkowo układ MAX232 jako konwerter poziomów. Porty ADC służą do odczytywania napięć występujących na poszczególnych pinach każdego z tych portów. Zakres pomiarowy wynosi 0-5V. Porty PD natomiast mogą być wykorzystane na przykład jako 4 bitowa szyna danych, lub też każdy z osobna. W naszym przypadku wykorzystujemy tylko jeden port do pomiaru temperatury czujnikiem DS18B20 gdzie komunikacja odbywa się cyfrowo.



Rysunek 10. Widok płytki mikroprocesora

5. Budowa serwera – sprzęt

System został zbudowany w oparciu o niezawodną wersję systemu operacyjnego LINUX DEBIAN w wersji 5.1 LENNY. Jest to stabilne rozwiązanie do tego typu przedsięwzięć. Minimalne wymagania sprzętowe to platforma oparta na procesorze Celeron o minimalnych wartościach: częstotliwość 433Mhz, pamięci ram 256MB oraz dysk twardy o pojemności 4,3GB. Taka konfiguracja pozwala na swobodne działanie serwera jak i przeliczania potrzebnych danych. Wymagany jest również port szeregowy COM oraz interfejs sieciowy z dostępem do sieci Internet.

ROZDZIAŁ 2 – Implementacja

I. Program działający w mikroprocesorze

Powiemy sobie więcej na temat programu pracującego w naszym procesorze. Środowisko programowania to język C. Do pisania oraz kompilowania programów najbardziej polecany jest program Programers Notepad oraz specjalny kompilator avr-gcc który z nim współpracuje. Kompilacja programu odbywa się poprzez plik Makefile w którym zawarte są podstawowe informacje takie jak prędkość, rodzaj mikroprocesora, format pliku wynikowego kompilacji – zazwyczaj jest to plik hex w formacie Intela którym programujemy mikroprocesor za pomocą odpowiedniego programatora ISP. Oto przykładowe wartości takiego pliku :

```
# nazwa MCU
MCU = atmega8

# Częstotliwość procesora.
F_CPU = 11059200

# Format wyjściowy. (srec, ihex, binary)
FORMAT = ihex
```

Do prawidłowego funkcjonowania procesora oraz konfiguracji portów niezbędne są poszczególne procedury. Zacznijmy może od podstawowej konfiguracji. Najpierw definiujemy porty które będziemy odpowiednio wykorzystywać a więc :

1. Port dla czujnika DS18B20

Aby zapewnić poprawną funkcjonalność czujnika, należy odpowiednio skonfigurować port, oraz zaimplementować funkcje wysyłające polecenia jak i również odbierające dane z czujnika.
Na początek definicja portu czujnika

```
// rejestry linii DQ
#define PIN1W PINB
#define DDR_DQ  DDRB
#define PORT_DQ PORTB
#define DQ      PB2
```

Pierwsza funkcja dla resetowania urządzeń 1 Wire

```
int ow_reset(void)
{
  int presence;
  delay(10);        // zaczekaj 10us
  CLR_DQ;           // ustaw stan 0
  delay(700);       // odczekaj 700us
  SET_DQ;           // ustaw stan 1
  delay(97);        // czekajna signal czujnika
  presence = PIN1W & (1 << DQ);   //PORT_DQ & ( 1 << 2);   // odbierz - jeśli
slave zrobil CLR_DQ to znaczy ze jest, stan wysoki tzn ze go nie ma
  delay(691);
return(presence);      // zwroc sygnal
}
```

Funkcja wysyłająca jeden bit do magistrali 1 Wire :

```
void write_bit(char bitval)
{
  if(bitval==1)   // zapis jedynki
  {
    CLR_DQ;      // linia DQ stan 0
    delay(5);   // 6us
    SET_DQ;      // ustaw DQ na 1
    delay(97);   // 70us, a moze byc od 59 do n/a us
  }
  else         // dla zapisu zera! wtedy inny delay!
  {
    CLR_DQ;      // linia DQ stan 0
    delay(97);   // 70us, a moze byc od 60 do 120us
    SET_DQ;      // ustaw DQ na 1
    delay(13);   // 10us, a moze byc od 8 do n/a us
  }
}
```

Aby móc wysłać 1 bajt piszemy tę oto funkcję :

```
void write_byte(uint8_t val)
{
  int loop;
  uint8_t temp;
  for ( loop=0 ; loop<8 ; loop++ )
  {
    temp = val>>loop;
    temp &= 0x01;
    write_bit(temp);
  }
}
```

Podobnie funkcje dla odczytu danych, pierwsza czytająca 1bit

```
uint8_t read_bit(void)
{
  uint8_t bit;
  CLR_DQ;              // ustaw 0
  delay(8);           // 6us, a moze byc od <5;15>us
  SET_DQ;              // ustaw 1
  delay(6);           // 5us, a moze byc <5;12>us
  bit = PIN1W & (1 << DQ);   //czytaj
  delay(83);           // 60us, a może być od 50 do n/a us

return(bit);           // zwroc warotść
}
```

Oraz funkcja czytająca 1 bajt :

```
uint8_t read_byte(void)
{
  int loop;
  uint8_t value;
  value  = 0x00;

  for ( loop=0 ;loop<8 ;loop++ )
  {
    if( read_bit() )
    value |= 0x01 << loop
    }
  return(value);
}
```

2. Porty analogowo-cyfrowe ADC

```
void adc_init(void)
{
    ADCSRA = _BV(ADEN)|_BV(ADPS0)|_BV(ADPS1)|_BV(ADPS2);
}
```

Gdzie ustawienie ADEN uruchamia przetwornik, a pozostałe bity konfigurują preskaler który dzieli częstotliwość procesora dla przetwornika ADC. Niestety im wyższa tym pomiar jest niedokładniejszy. Ustawienie wszystkich 3 bitów na „1” konfiguruje nam preskaler na wartość „128”. Oczywiście dostępne są wartości 2, 4, 8, 16, 32, 64, 128.

Jak wspominałem już, do prawidłowego funkcjonowania czujników DS18B20 musimy odpowiednio skonfigurować środowisko. Czujniki te są bardzo czułe na czasy operacji, dlatego też skonfigurowany został osobno timer dla opóźnień.
Odpowiadają za to rejestry TCCR gdzie również mamy możliwość ustawienia preskalera ale tym razem jego wartości możemy dobrać z wartości 8, 64, 256, 1024.

3. Port UART – RS232

Oczywiście konfigurujemy też port UART do komunikacji poprzez RS232.
Wygląda to mniej więcej tak :

```
void USART_Init( unsigned int baud )
{
  /* ustawienie szybkości transmisji */
  UBRRH = (F_CPU/(baud*16L)-1) >> 8;
  UBRRL = (unsigned char)(F_CPU/(baud*16L)-1);
  /* Uruchomienie odbiornika oraz nadajnika */
  UCSRB = (1<<RXCIE)|(1<<TXEN)|(1<<RXEN);
    /* Ustawienie ramki na: 8data bit, 2stop bits */
  UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}
```

Z tak skonfigurowanym mikroprocesorem przystępujemy do pracy.
Oto podstawowe funkcje które pobierają nam wartości.
Pierwsza z nich to pobieranie danych z przetwornika analogowo-cyfrowego.
Przetwornik ten jest 10 bitowy a wiec maksymalna zwracana wartość to 1023.

```
unsigned int adc_read_channel(uint8_t chn)
{
    unsigned int t;
    ADMUX=chn;  //wybór kanału
    ADCSRA |= _BV(ADSC);       // zezwolenie na konwersję
    while(bit_is_set(ADCSRA,ADSC))  // // Oczekiwanie na zakończenie przetwarzania
    {};
    t=ADC;
    return t;
}
```

Funkcję tą wywołujemy z parametrem który definiuje port z którego wartość ma być odczytana. Zwracana jest wartość przetwornika z zakresu 0-1023.
ADMUX odnosi się do konkretnego kanału od 0 do 7. My będziemy wykorzystywać tylko pierwsze cztery.
Oczywiście mamy też funkcje służące do komunikacji poprzez port UART.

```
void USART_Transmit( unsigned char data )
{
  /* Zaczekaj aż buffor wysyłania będzie pusty*/
  while ( !( UCSRA & _BV(UDRE)) )
  ;
  /* umieść dane w buforze , oraz wyślij */
  UDR = data;
}

unsigned char USART_Receive( void )
{
```

```
  /* Czekaj na dane w buforze */
  while ( !(UCSRA & (1<<RXC)) )
  ;
  /* zwróć odebrane dane */
  return UDR;
}
```

Sercem programu jest pętla wykonująca się w nieskończoność, jeżeli na wejściu RS232 pojawi się liczba „1” zostanie ona wykonana.

```
for(;;){
      if(USART_Receive()=='1'){
            Timer_Config_ns(); // konfigurowanie timera na nano sekundy
            Read_Temperature();
            Read_Pressure();
            Read_hum();
            Read_sol();
            }
   }
```

Każda z funkcji z prefixem Read sczytuje nam określoną wartość z poszczególnych czujników, a są to temperatura, ciśnienie, wilgotność oraz nasłonecznienie. W sekcji Najciekawsze fragmenty kodów są przedstawione metody obliczeń.

4. Najważniejsze fragmenty kodów mikroprocesora

a) Odczyt temperatury

```
ow_reset();
write_byte(0xCC);
```

Skip ROM, Komenda pozwalająca na zaadresowanie wszystkich układów na magistrali jednocześnie i wysłaniu komendy do wszystkich jednocześnie

```
write_byte(0x44);
```

Start Conversion, Komenda inicjalizująca konwersję temperatury

```
delay_5ms(75);delay_5ms(75);delay_5ms(75);   // Czas na konwersje
```

```
ow_reset();
write_byte(0xCC);    // Skip ROM
write_byte(0xBE);
```

Read Scratch Pad, komenda pozwalająca na odczyt pamięci
(wszystkich 9 bitów). Odczyt może być przerwany w dowolnym momencie przez impuls reset

```
USART_write_text("t<");
USART_write_text(dtostrf(temp,0,2,buffer));
USART_write_text(">");
```

Przesłanie danych przez port UART w postaci t<wartość>

b) Odczyt ciśnienia

```
a=adc_read_channel(0);
a = a * 0.0048828125;
cisnienie = 217.86483333333333333 * (((a-1)/8.77193)+3.731) +
150.00036333333333;
USART_write_text("c<");
USART_write_text(dtostrf(cisnienie,0,2,buff));
USART_write_text(">");
```

Najpierw pobieramy wartość kanału 0 adc do którego podłączony jest czujnik nasłonecznienia. Wartość naszego napięcia odniesienia wynosi 5V a więc przeliczamy wynik na Volty w drugiej linii kodu.
Następnie zostaje wyliczone ciśnienie, ze względu na mniejszy próg błędu wartości nie zostały przybliżone. Zakres samego czujnika jest w przedziale 3,6 – 4,2V wiec dla dokładniejszego pomiaru ten zakres został rozciągnięty wzmacniaczem operacyjnym to wartości 0-5V. Liczba 217.86483 to średnie wzmocnienie układu pomnożona przez offset wzmacniacza operacyjnego a następnie dodana zostaje dolna granica czujnika 15kPa.
Na końcu wynik przesyłany jest poprzez port UART, funkcja „dtostrf" konwertuje dane z postaci double do postaci ASCII z dokładnością zdefiniowaną do 2 miejsc po przecinku.

c) Odczyty wilgotności oraz nasłonecznienia

Są to akurat podobne funkcje ponieważ pobierają tylko wartość napięcia. Różnica polega tylko na wyborze portu ADC

```
a=adc_read_channel(1);
a = a * 0.0048828125;
USART_write_text("h<");
USART_write_text(dtostrf(a,0,3,buff));
USART_write_text(">");
```

W ten sposób do komputera docierają dane jak przedstawiłem poniżej :
c<1022.18>h<1.602>s<0.000>t<3.13>

II. Skrypty działające na serwerze

Jak już wspomnieliśmy w poprzednim punkcie, cron uruchamia poszczególne skrypty w określonej kolejności. A wiec zaczniemy od pierwszego.

1. reset

Skrypt ten ma za zadanie wyczyszczenie zawieszonych procesów. Jeżeli w przeciągu 30 sekund nie zostaną zakończone procesy write.php oraz rs232 wtedy skrypt usuwa je aby następne mogły się wykonać prawidłowo. Głownie chodzi tutaj o skrypt „rs232" ponieważ czasami pomiar może być czytany dłużej lub też mogą wystąpić jakieś błędy. Oczywiście jak widzimy widnieje także zapis logów.

```
echo "Resetowanie :   "`date` >> /var/www/meteo/site/scripts/log
sleep 30
killall write.php
killall rs232
```

Rysunek 11 Fragment kodu skryptu "reset "

Po tej operacji czas na kolejną, jest nią pobieranie danych z mikrokontrolera.

2. write.php

Jest to przysłowiowe serce komunikacji serwera z elektroniką. Skrypt odpowiada za pobieranie danych z portu RS232 oraz zapis do bazy MYSQL w odpowiednie tabele.
Pierwszą rzeczą jest pobranie z pliku danych do dostępu do bazy danych.
Następnie pobierane są dane z mikrokontrolera. Program rs232 wysyła do mikrokontrolera polecenie przesłania danych w postaci „1" z prędkością 9600 bodów, następnie odczekuje 3 sekundy na odbiór , i pobierane są 34 bity oraz zapisanie do pliku data.dat.

```
do{
system("rm /var/www/meteo/database/data.dat");
system("rs232 -w 3 -b9600 -p8n1 -s'1' -r34 >> /var/www/meteo/database/data.dat");
$data = file_get_contents('/var/www/meteo/database/data.dat');
sscanf($data,"t<%e>c<%e>h<%e>s<%e>",$temp,$press,$hum,$sol);
}while($temp == 0 || $press == 0 || $hum == 0 || $sol == 0);
```

Rysunek 12 Fragment kodu skryptu "write.php"

Przykładowa zawartość pliku data.dat „t<6.25>c<1015.27>h<1.260>s<0.010>”
Dane zostają wczytane do poszczególnych zmiennych dzięki funkcji sscanf która wybiera z danego stringa poszczególne wartości oznaczone jako %e i ładuje je do zmiennych zdefiniowanych dalej. Jeżeli któraś z wartości będzie źle pobrana pętla do – while ponowi procedurę. Pobrane dane zostają zapisane do bazy danych.

3. safir3000

Jest to rozbudowa istniejącego już systemu dostępnego w serwisie pogodynka.pl. Rozbudowa polega na pobieraniu określonej (definiowanej) Ilości obrazków z mapą, a następnie zostaje utworzony animowany obrazek. Dzięki temu można lepiej interpretować mapę i wywnioskować kierunek komórki burzowej. Całe działanie polega na pobraniu określonej ilości obrazków. Skrypt oczywiście posiada obsługę rotacji plików, to znaczy jeżeli zostanie przekroczona dopuszczalna liczba obrazków, to najstarszy zostaje kasowany. Do tworzenia animacji użyty został program „Convert” z pakietu Image-Magic który dostępny jest w repozytoriach Debian-a.

4. generate_all

Jest to skrypt generujący wykresy w programie gnuplot. Pierwszym zadaniem tego skryptu jest wyczyszczenie starych plików przed utworzeniem nowych. Kasowane są pliki z katalogów „data” w którym przechowywane są pliki z danymi oraz z katalogu „img” gdzie znajdują się pliki z wykresami.
Następnym krokiem jest wygenerowanie nowych plików danych dla wykresów za pomocą skryptu php dla każdego z pomiarów. Uruchamia się on z parametrem (ilością pobranych danych) odpowiednio dla 4,12 i 24 godzin.

```
#generowanie plikow danych dla gnuplota 4,12,24h
/var/www/meteo/site/scripts/php/gen_high_data.php 240
/var/www/meteo/site/scripts/php/gen_press_data.php 240
/var/www/meteo/site/scripts/php/gen_sol_data.php 240
/var/www/meteo/site/scripts/php/gen_temp_data.php 240

/var/www/meteo/site/scripts/php/gen_high_data.php 720
/var/www/meteo/site/scripts/php/gen_press_data.php 720
/var/www/meteo/site/scripts/php/gen_sol_data.php 720
/var/www/meteo/site/scripts/php/gen_temp_data.php 720

/var/www/meteo/site/scripts/php/gen_high_data.php 1440
/var/www/meteo/site/scripts/php/gen_press_data.php 1440
/var/www/meteo/site/scripts/php/gen_sol_data.php 1440
/var/www/meteo/site/scripts/php/gen_temp_data.php 1440


#generowanie rysunkow dla 4,12,24h
/var/www/meteo/site/scripts/gplot/./last4h
/var/www/meteo/site/scripts/gplot/./last12h
/var/www/meteo/site/scripts/gplot/./last24h
```

Rysunek 13 Fragment skryptu " generate_all"

Wszystkie skrypty php generujące dane, to nic innego jak zapis wartości z bazy danych do poszczególnych plików txt.

Przykładowa zawartość jednego z plików :

---

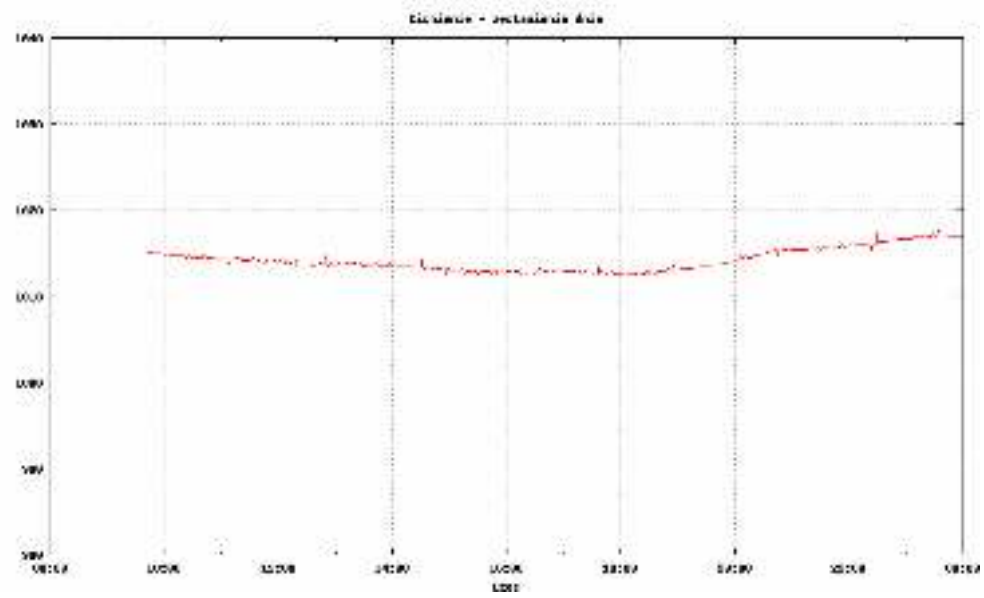
data	godzina	wilgotnosc
2009-05-05	22:19:04	64.89
2009-05-05	22:18:05	65.55
2009-05-05	22:17:04	66.82
2009-05-05	22:16:04	74.63
2009-05-05	22:15:07	65.22
2009-05-05	22:14:05	66.49
2009-05-05	22:13:05	64.22

Po tej operacji zostają wygenerowane wykresy dla poszczególnych okresów czasu.

```
set terminal png size 1000,600
set title "Wilgotnosc powietrza - ostatnie 4 godziny"
set grid
unset key
set style data lines
set output '/var/www/meteo/site/img/hig_4.png'
set xdata time
set autoscale y
set timefmt "%H:%M:%S"
set format x "%H:%M"
set xlabel "Czas"
set xtics 1800
set yrange [*:*]
plot "/var/www/meteo/site/data/meteo_hig_240.txt" using 2:3

set title "Cisnienie atmosferyczne [ 330 m.n.p.m] ostatnie 4 godziny"
set output '/var/www/meteo/site/img/press_4.png'
set yrange [*:*]
plot "/var/www/meteo/site/data/meteo_press_240.txt" using 2:3

set title "Naslonecznienie - ostatnie 4 godziny"
set output '/var/www/meteo/site/img/sol_4.png'
set yrange [*:*]
plot "/var/www/meteo/site/data/meteo_sol_240.txt" using 2:3

set title "Temperatura powietrza - ostatnie 4 godziny"
set output '/var/www/meteo/site/img/temp_4.png'
set xzeroaxis linetype 3 linewidth 5
set yrange [*:*]
plot "/var/www/meteo/site/data/meteo_temp_240.txt" using 2:3
```

Rysunek 14 Fragment skryptu "last4h"

Kod jest w miarę prosty i czytelny. Jak widzimy zostają zdefiniowane między innymi:
W linii 2 tytuły dla wykresów, linia 3 - siatki, linia 4 - styl wykresu, linia 5 -plik wynikowy, od linii 6 do 12 opisy osi X oraz Y,
jak i również częstość podpisów osi w linii 13. Dane są pobierane z pliku wygenerowanego przez poprzedni skrypt. Pobierana jest 2 i 3 kolumna pliku odpowiednio dla X i Y.



Rysunek 15 Przykładowy plik wynikowy - Wykres ciśnienia

Tak wygląda przykładowy plik wynikowy skryptu dla ciśnienia atmosferycznego.

5. archiver

Tutaj wykonują się operacje archiwizacji wykresów z każdego dnia.
Jak widzieliśmy w pliku crontab skrypt ten wykonywany jest pięć minut po północy tak aby można było archiwizować cały poprzedni dzień.

```
#!/bin/sh
echo "Archiwizacja .. "`date` >> /var/www/meteo/site/scripts/log
/var/www/meteo/site/scripts/php/./archive.php
/var/www/meteo/site/scripts/gplot/./archive_plot
day=`date +%Y-%m-%d --date="1 day ago"`

mkdir /var/www/meteo/site/img/archive/$day
mv /var/www/meteo/site/img/archive/archive_hig.png /var/www/meteo/site/img/archive/$day/hig_day.png
mv /var/www/meteo/site/img/archive/archive_press.png /var/www/meteo/site/img/archive/$day/press_day.png
mv /var/www/meteo/site/img/archive/archive_temp.png /var/www/meteo/site/img/archive/$day/temp_day.png
mv /var/www/meteo/site/img/archive/archive_sol.png /var/www/meteo/site/img/archive/$day/sol_day.png
```

Rysunek 16 Fragment skryptu "archiver"

Archiwizacja odbywa się poprzez wykonanie skryptu archive.php który zrzuca nam dane z bazy mysql do pliku tekstowego, a następnie wykonywany jest archive_plot który rysuje wszystkie wykresy. Później pobierana jest dana którą trzeba cofnąć o dzień, ponieważ jak wiemy skrypt wykonuje się po północy. Na koniec kopiowanie plików wynikowych do odpowiedniego katalogu nazwanego wcześniej datą.

6. gen_data_now

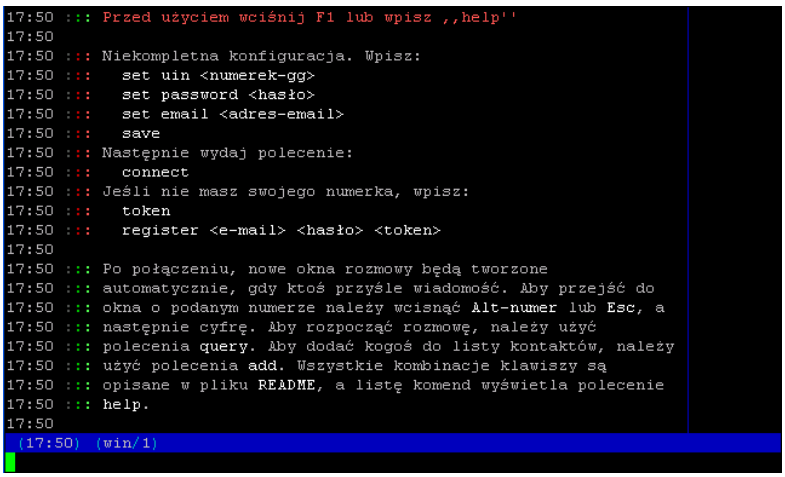
Ostatni skrypt odpowiada za wysyłanie bieżących danych pogodowych dla bot-a gadu-gadu w postaci pliku tekstowego. Są to wartości bieżące oraz maksymalne.

Przykładowy plik wynikowy dla wartości maksymalnych:
„2009-05-09 11:31:07,96.13,2009-05-09 13:43:06,96.26,2009-05-09 14:20:04,26.06,2009-05-09 13:38:05,1022.18”

Przykładowy plik wynikowy dla wartości bieżących :
„21.83,90.42,24.56,1021.09,0.938”

7. Bot Gadu-Gadu

Oczywiście aby ułatwić użytkownikom dostęp do danych został opracowany odpowiedni skrypt/bot gadu-gadu. Jako klient dostępu do sieci został wykorzystany dobrze znany program EKG (Eksperymentalny Klient Gadu-Gadu) działający na platformie linux.



Rysunek 17 Okno programu EKG

Zasada prezentacji danych jest przedstawiana na dwa sposoby, poprzez opis zmieniany co 20sekund, oraz poprzez wysłanie dowolnego znaku lub ciągu znaków rozpoczynających rozmowę.

Sam program nie wymaga praktycznie żadnych ustawień poza wpisaniem odpowiedniego numeru gg oraz hasła.

Zasada działania jest następująca, program EKG uruchamiamy:

“screen ekg -c /root/.gg/ekg-pipe”

Screen to aplikacja pozwalająca na uruchamianie innych aplikacji w jednej sesji terminala. Następnie tworzony jest plik potoku tak aby można było wysyłać informacje do klienta EKG z innych skryptów. Opiszę teraz 2 główne konfiguracje klienta.

a) dodanie timera dla opisów

„timer -a last */20 exec /var/www/meteo/site/scripts/botgg/./now”

Co 20 sekund zostaje uruchomiony skrypt “now” który wysyła poprzez potok do klienta dane do opisu z bieżącymi informacjami pogodowymi.

```
#!/bin/sh
LAST=`/usr/bin/php -q /var/www/meteo/site/scripts/botgg/gg.php`
echo "/away "$LAST > /root/.gg/ekg-pipe
```

Rysunek 18 Fragment kodu timera dla bota Gadu Gadu

Oczywiście jak wiemy opisy w gadu-gadu mają limit określonych znaków, tak więc skrypt gg.php selekcjonuje informacje pogodowe na 2 wartości oraz krótkie info w zależności od ilości sekund bieżącej minuty. Wykonywanie skryptu co 20 sekund pozwala na selekcje informacji poprzez warunki IF.

```
#!/usr/bin/php
<?
$s=exec("date +%S");
$file = file("/var/www/meteo/site/data/now/meteo_now.txt");
foreach($file as $value) {
$exp = explode(",",$value);
}

if($s>0 && $s<15 || $s>30 && $s<45)
    {
    echo "Temperatura: ".$exp[2]." C Cisnienie: ".$exp[3]." hPa";
    }

elseif ($s>30 && $s<45 || $s>15 && $s<30)
    {
    echo "Wilgotnosc: ".$exp[0]." % Naslonecznienie: ".$exp[1]."%";
    }

elseif($s>45 && $s<59)
    {
    echo "Napisz cos do mnie - dowiesz sie wiecej .. :)";
    }
?>
```

Rysunek 19 Skrypt php zwracający odpowiedni opis dla bota

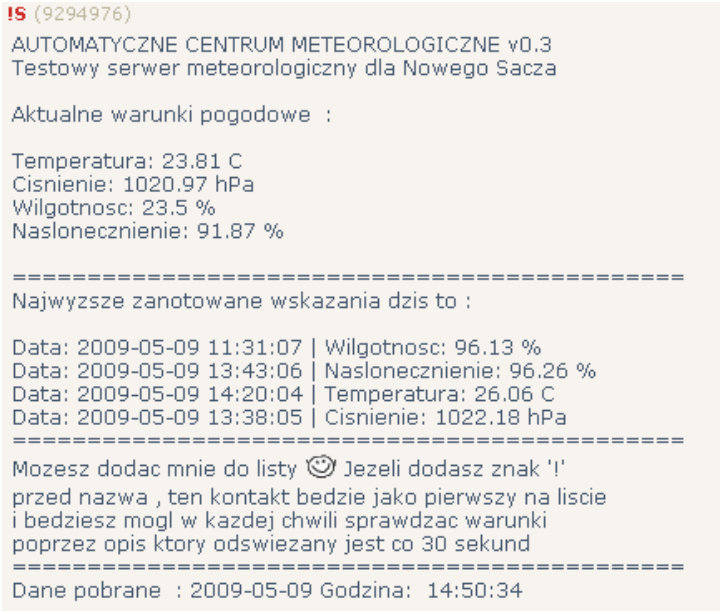
Wynik działania skryptu z opisem :

Rysunek 20 Przykładowy stan opisu bota Gadu-Gadu

b) dodanie zdarzenia dla przychodzącej wiadomości

/on --add msg,chat * /exec --bmsg %1
/var/www/meteo/site/scripts/botgg/./mow %1 '%4'"

Dodaje nam zdarzenie które zostaje uruchomione po otrzymaniu wiadomości od dowolnego użytkownika gadu-gadu. Uruchamia nam polecenie systemowe w postaci skryptu „mow" i odsyła jego wynik w jednej wiadomości.
Wynik działania jest następujący :



Rysunek 21 Wynik działania bota Gadu-Gadu

A oto skrypt „mow”

```
#!/bin/sh
date=$(date +%Y-%m-%d)
date2=$(date +%H:%M:%S)
echo "AUTOMATYCZNE CENTRUM METEOROLOGICZNE v0.3"
echo "Testowy serwer meteorologiczny dla Nowego Sacza"
echo ""
echo "Aktualne warunki pogodowe  :"
echo ""
echo `/var/www/meteo/site/scripts/botgg/pokaz_dane.php 1`
echo `/var/www/meteo/site/scripts/botgg/pokaz_dane.php 2`
echo `/var/www/meteo/site/scripts/botgg/pokaz_dane.php 3`
echo `/var/www/meteo/site/scripts/botgg/pokaz_dane.php 4`
echo ""
echo "=================================================="
echo "Najwyzsze zanotowane wskazania dzis to :"
echo ""
echo `/var/www/meteo/site/scripts/botgg/pokaz_dane.php 11`
echo `/var/www/meteo/site/scripts/botgg/pokaz_dane.php 21`
echo `/var/www/meteo/site/scripts/botgg/pokaz_dane.php 31`
echo `/var/www/meteo/site/scripts/botgg/pokaz_dane.php 41`
echo "=================================================="
echo "Mozesz dodac mnie do listy :) Jezeli dodasz znak '!' "
echo "przed nazwa , ten kontakt bedzie jako pierwszy na liscie"
echo "i bedziesz mogl w kazdej chwili sprawdzac warunki"
echo "poprzez opis ktory odswiezany jest co 30 sekund"
echo "=================================================="
echo "Dane pobrane  : "`echo $date`" Godzina: " `echo $date2`
```

Rysunek 22 Kod źródłowy skrypu bota "mow"

Pokaz_dane.php wywołany z parametrem zwraca nam z bazy danych żądaną wartość bieżącą dla prefiksów 1,2,3,4 oraz maksymalną dla 11,21,31 i 41

ROZDZIAŁ 3 - Oprogramowanie oraz praca serwera

Praca serwera ma za zadanie archiwizację wszystkich danych, prezentowanie ich w postaci wykresów, oraz również poprzez znany komunikator gadu-gadu. W pierwszej kolejności zajmiemy się wdrożeniem oprogramowania.

1. Wdrożenie oprogramowania

Aby system mógł poprawnie funkcjonować należy zainstalować określone oprogramowanie.

a) Gnuplot

Jest to aplikacja służąca do generowania wykresów.
Instalacja : poprzez polecenie „apt-get install gnuplot”

b) Image-Magic

Aplikacja modyfikująca pliki graficzne.
Instalacja : poprzez polecenie „apt-get install imagemagic”

c) EKG

Klient sieci Gadu-Gadu.
Instalacja: poprzez polecenie „apt-get install ekg”

d) Phpsysinfo

Aplikacja umożliwiająca podgląd danych dotyczących systemu.
Instalacja: poprzez polecenie „apt-get install phpsysinfo”

e) Sjinn

Oprogramowanie służące do komunikacji poprzez porty RS232
Instalacja: pobieramy darmową wersję ze strony http://sjinn.sourceforge.net/
Po rozpakowaniu, kompilujemy poprzez polecenie „make”
Po skończonej kompilacji, dodajemy nasz program do poleceń systemowych poleceniem „cp rs232 /usr/local/bin”

2. Baza danych MySql

Przechowywaniem wszystkich danych dotyczących pogody zajmuję się baza danych. Co jedną minutę zostają zapisane bieżące wartości pogodowe pobrane przez odpowiedni skrypt który komunikuje się z procesorem poprzez port RS232.

Struktura bazy jest następującej postaci :
- 4 tabele po 2 kolumny (data, wartość) dla każdego czujnika

cisnienie		
Pole	***Atrybuty***	***Domyślnie***
data [timestamp]	ON UPDATE CURRENT_TIMESTAMP	CURRENT_TIMESTAMP
cisnienie_alt [float]	UNSIGNED	

Tabela 4 Struktura tabeli cisnienie

naslonecznienie		
Pole	***Atrybuty***	***Domyślnie***
data [timestamp]	ON UPDATE CURRENT_TIMESTAMP	CURRENT_TIMESTAMP
naslonecznienie [float]	UNSIGNED	

Tabela 5 Struktura tabeli naslonecznienie

temp_out		
Pole	**Atrybuty**	**Domyślnie**
data [timestamp]	ON UPDATE CURRENT_TIMESTAMP	CURRENT_TIMESTAMP
temp [float]	UNSIGNED	

Tabela 6 Struktura tabeli temp_out

wilgotnosc		
Pole	**Atrybuty**	**Domyślnie**
data [timestamp]	ON UPDATE CURRENT_TIMESTAMP	CURRENT_TIMESTAMP
wilgotnosc [float]	UNSIGNED	

Tabela 7 Struktura tabeli wilgotność

3. Zarządzanie skryptami.

Plik który zarządza całą pracą znajduje się zazwyczaj w /etc/crontab
I w naszym przypadku wygląda on tak :

```
# m h  dom mon dow   command
*/1 * * * * root /var/www/meteo/site/scripts/./reset
*/1 * * * * root /var/www/meteo/database/./write.php
*/10 * * * * root /var/www/meteo/site/scripts/safir/./safir3000
*/5 * * * * root /var/www/meteo/site/scripts/./generate_all
5 0 * * *   root /var/www/meteo/site/scripts/./archiver
*/1 * * * * root /var/www/meteo/site/scripts/php/./gen_data_now.php
```

Skrypty te wykonują się co określony czas i tak np. */1 wykonuje się dokładnie co jedną minutę, natomiast w przypadku archiver-a skrypt ten wykonuje się pięć minut po północy.

4. Serwis WWW i jego składniki

Jest to prosta strona internetowa która ma na celu wyświetlanie wyników stacji pomiarowej w postaci wskaźników flash, plików graficznych - wykresów jak i również są tam zebrane potrzebne informacje z różnych innych serwisów takie jak prognozy, zdjęcia satelitarne.

Rysunek 23 Wygląd strony internetowej

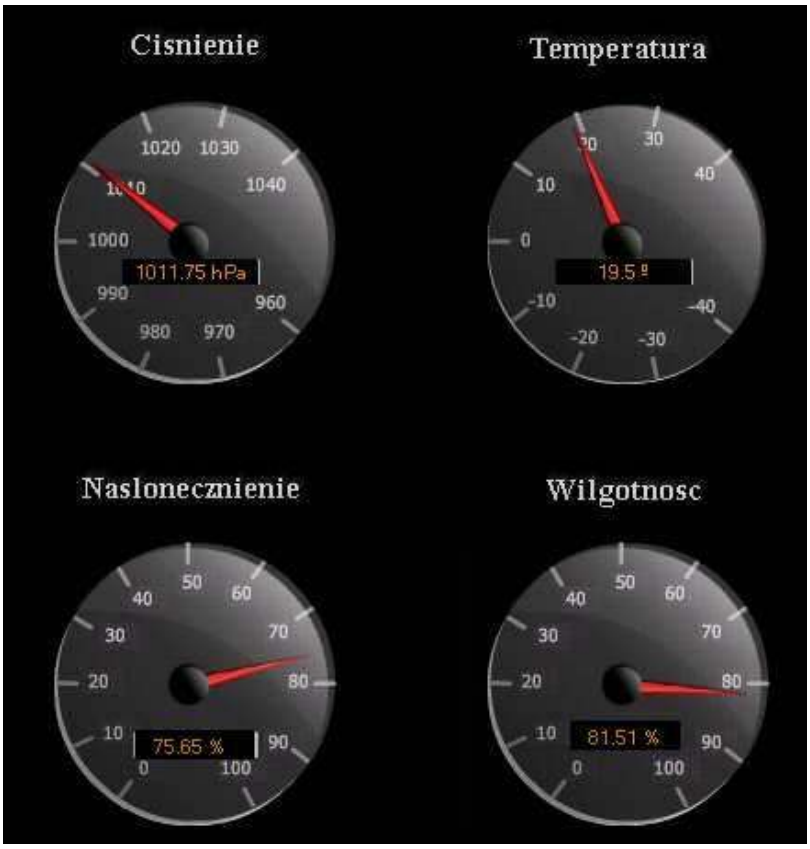
a) Informacje

Tutaj wyświetlane są informacje dotyczące samego serwera oraz warunków jego pracy. Wykorzystany został program „phpsysinfo” po lekkich modyfikacjach – kolor, styl.

Rysunek 24 Screen strony "Informacje o Systemie"

b) Pogoda

Prezentowane na tej stronie wskazania pochodzą wprost ze stacji pomiarowej i są one wyświetlane za pomocą czterech wskaźników flash. Informacje pogodowe są odświeżane co 1 minutę natomiast same wskaźniki co 20 sekund, tak aby nie dopuścić do pomijania pomiarów.



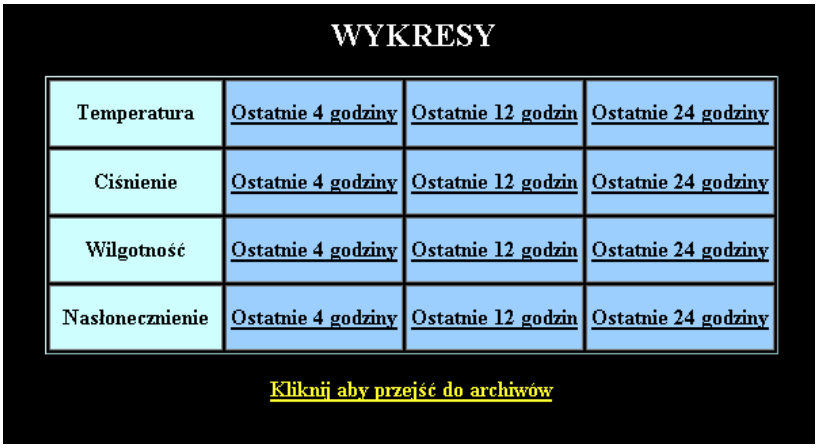
Rysunek 25 Przykładowe wartości prezentowane przez wskaźniki flash

c) Burze

Jak już wcześniej wiemy, na serwerze działa skrypt safir3000, tutaj właśnie prezentowany jest jego plik wynikowy - animacja „burze.gif”.

d) Wykresy

Ta strona pozwala nam na przegląd generowanych wykresów z 4,12 i 24 godzin dla każdego pomiaru oraz archiwizowane dane.



Rysunek 26 Screen strony "Wykresy"

e) Prognozy

Jest to dopełnienie serwisu poprzez prezentacje meteogramów, które udostępnia serwis http://new.meteo.pl. Jest to najbardziej sprawdzająca się prognoza dostępna w Internecie.

f) Satelita

Tutaj również natrafimy na dopełnienie w postaci animacji zdjęć satelitarnych z ostatnich dwóch godzin. Źródło danych pochodzi ze strony www.sat24.com

Podsumowanie

Prognoza pogody jest nieodłącznym elementem wiadomości, jakie zdobywamy i wykorzystujemy w codziennym życiu. Oprócz tego, że jest to dobry temat rozmów, jeśli innego nie mamy, to również dowiadujemy się, co danego dnia na siebie włożyć, czy wziąć parasol i czego możemy się spodziewać za oknem, gdy planujemy czas wolny. Na całym globie codziennie tysiące ludzi zajmuje się sprawami związanymi z prognozowaniem pogody. Pracują stacje naziemne, w niebo ulatują balony z radiosondami, krążą samoloty, a nad tym wszystkim "czuwają" satelity meteorologiczne. Uzyskane w ten sposób dane trafiają do światowych centrów (Moskwa, Melbourne, Washington), a następnie krajowych biur pogodowych.

Głównymi atutami tego projektu jest ogólno dostępność podobnych pomiarów dla innych użytkowników. Jak wiemy dobrze, koszt dobrej klasy domowej stacji meteorologicznej przekracza niekiedy kwotę 500zł lub więcej, w tym wypadku za darmo każdy może mieć dostęp do stacji za pośrednictwem strony WWW lub też bot-a Gadu-Gadu który oczywiście jest wygodniejszy w użyciu, a większość użytkowników Internetu korzysta właśnie z tej sieci komunikatora. System również loguje dane pogodowe do późniejszej interpretacji, a także prezentuje zebrane dane z innych serwisów dla większej wygody użytkowników. Jedną z lepszych prezentacji jest animacja burz, przy pomocy której możemy nawet na dwie godziny wcześniej stwierdzić czy musimy obawiać się wyładowań atmosferycznych nad naszymi głowami. Stację tę oczywiście można rozbudować o kolejne czujniki na przykład deszczomierz, wiatromierz oraz wiele innych, a także dedykowany dla miasta Nowy Sącz, system ostrzegania przed burzami.

Bibliografia

Hubert Dróżdż, Joanna Drożdż - Skrypty w Shellu. Programowanie w powłoce Bash

PHP i MySQL. Tworzenie stron WWW. Wydanie drugie. Vademecum profesjonalisty

http://debian.linux.pl/ - w poszukiwaniu odpowiedzi dotyczących systemu linux korzystałem z forum (2009)

www.kurshtml.boo.pl/ - wskazówki dotyczące HTML (2009)

www.flashzone.pl/ - w poszukiwaniu rozwiązań problemów z flash-em. (2009)

www.elektroda.pl – pomoce dotyczące programowania mikrokontrolerów. (2009)

Spis Tabel

Spis rysunków