Politechnika Wrocławska

Projekt

Architektura Komputerów 2

System lokalizacji GPS z wykorzystaniem komputera Raspberry Pi 2

Authors:
Rafał Pieniążek
Jakub Pomykała

Supervisor:
Dr inż. Jędrzej Ułasiewicz

3 czerwca 2016

1 Wstęp

1.1 Cel projektu

Zaprojektować system określania pozycji z wykorzystaniem modułu GPS i komputera Raspberry PI 2. System ma wyświetlać pozycję na wyświetlaczu LCD.

1.2 Etapy projektu

- Zapoznanie się z komputerem Raspberry PI 2
- Zapoznanie się z układem lokalizacji satelitarnej GPS Global TOP FGPMMOSL3
- Zaprojektowanie połączenia komputera z modułem lokalizacji, wyświetlaczem LCD.
- Opracowanie programu obsługi lokalizatora i wyświetlacza
- Testy

1.3 Struktura kodów *NMEA*

NMEA jest protokołem komunikacji wykorzystywanym głownie w nawigacji morskiej. Dane przesyłane są w postaci sekwencji kodów oddzielonych przecinkami. Każde zdanie zawiera informacje m.in. o identyfikatorze wiadomości, czasie UTC, długości i szerokości geograficznej (wraz z identyfikatorami półkuli północnej/południowej oraz wschodniej/zachodniej), ilości satelitów użytych do ustalenia położenia. Poniżej przedsationo przykładowy kod:

```
$GPGLL,2305.91626,N,12017.06438,E,064951.00,A,A*61
```

1.4 Odczyt danych z modułu GPS

Po podłączeniu modułu lokalizacji do komputera Raspberry Pi poprzez złącze szeregowe możliwe jest odczytywanie danych w sposób analogiczny do czytania z pliku specjalnego. W związku z tym użycie komendy bashowej:

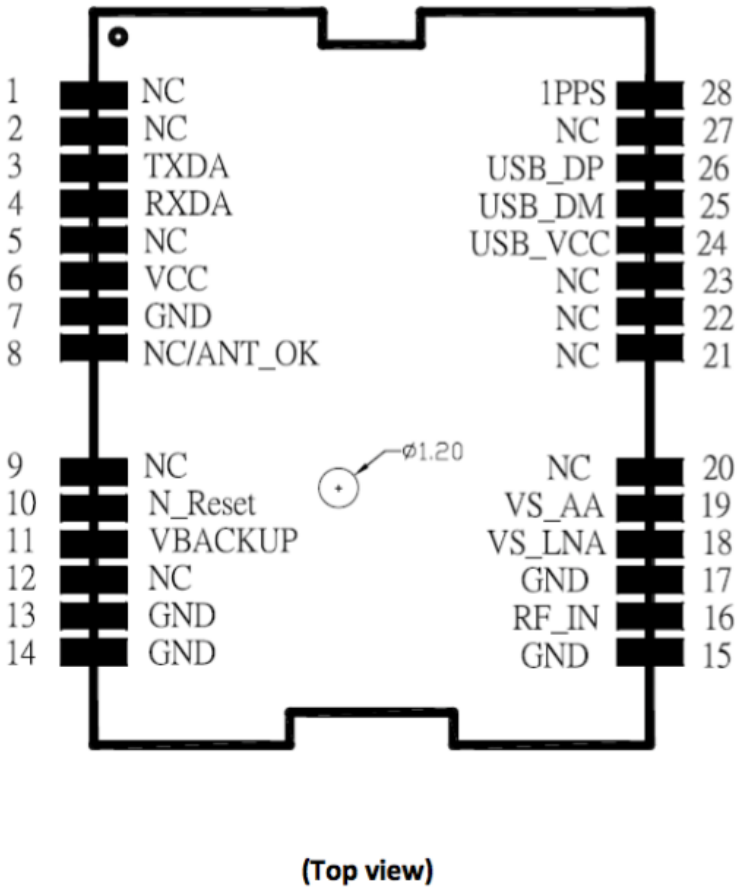
```
cat /dev/ttyAMA0
```

pozwoliło na wypisanie danych przesyłanych przez GPS na ekran konsoli. Odczyt pliku wymaga praw *roota*.

2 Opis projektu

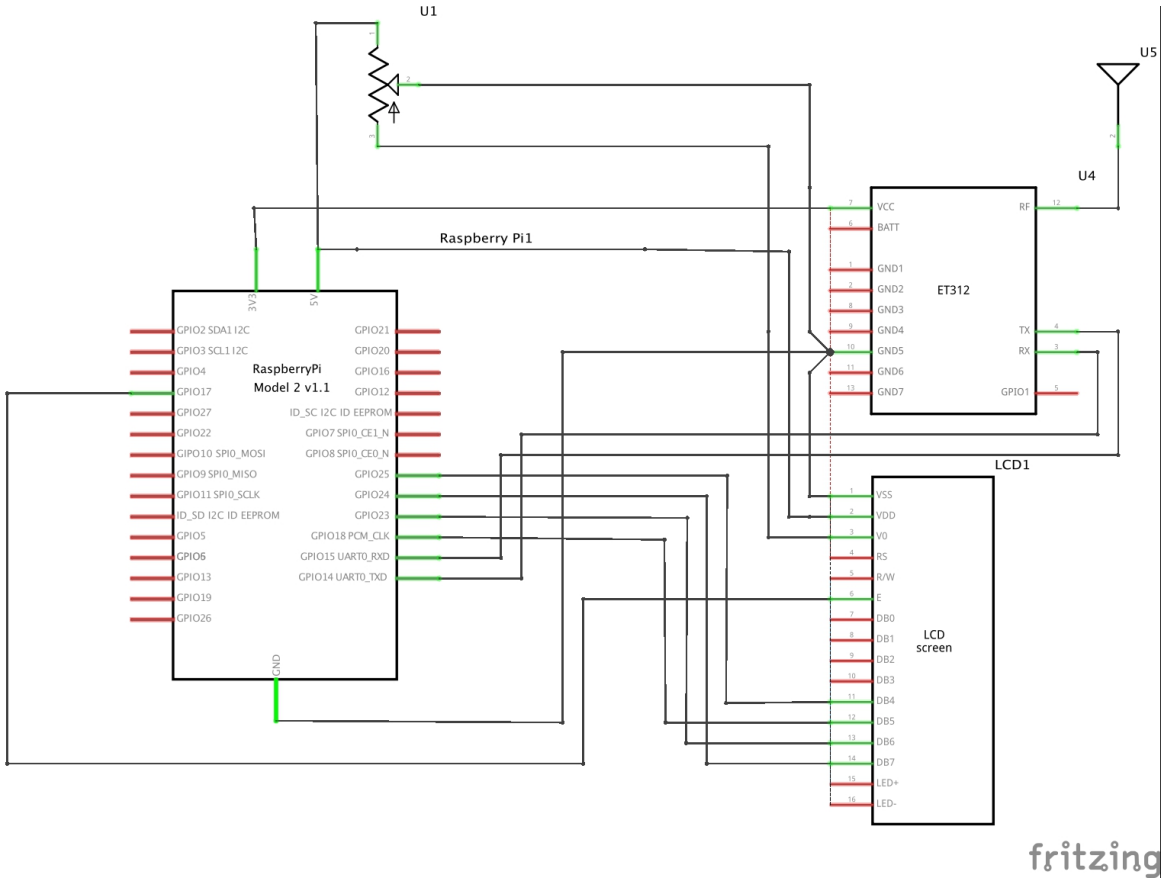
2.1 Wykonanie połączeń układu

Pierwszym etapem prac było zapoznanie się z dokumentacją techniczną modułu lokalizacji. Po przeanalizowaniu odpowiednich wyprowadzeń rozpoczęto fizyczny montaż modułu do płytki prototypowej. Podczas wykonywania tych czynności napotkano techniczny problem wykonania połączeń lutowanych. Mały rozmiar wyprowadzeń oraz niestandardowy rozmiar uniemożliwił użycia jakiejkolwiek podstawki do zamocowania w płytce prototypowej. Konieczne było ręczne przylutowanie wyprowadzeń przy zastosowaniu precyzyjnej lutownicy. Sprzęt jakim dysponowano był nieodpowiedniej klasy i mocy, istniało ryzyko przegrzania i uszkodzenia układu. Problem został rozwiązany dzięki wykorzystaniu sprzętu Politechniki Wrocławskiej. Podłączenie kolejnych elementów układu nie sprawiło większych problemów. Poniżej przedstawione zostały schematy połączeń, oraz wizualizacja układu.

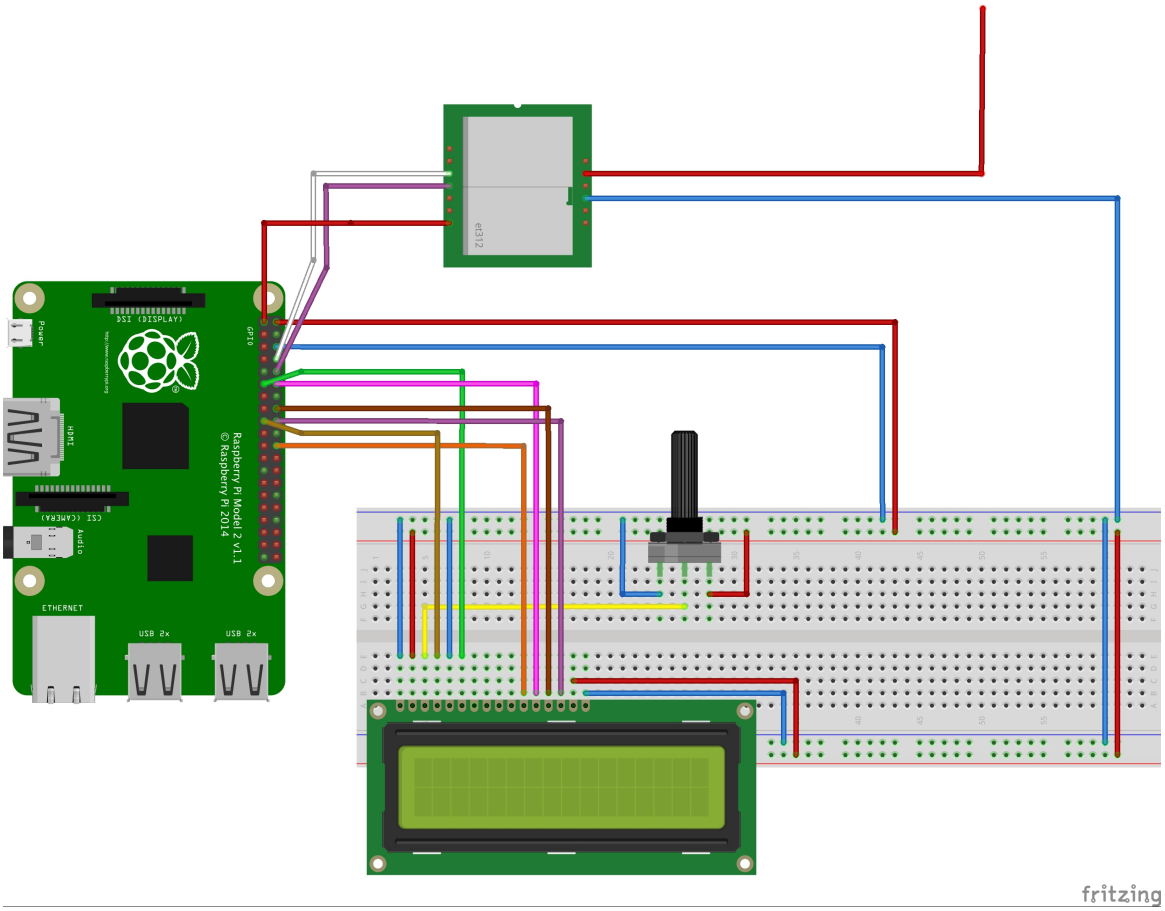


Rysunek 1: Schemat układu GPS z dokumentacji

Układ GPS do poprawnego działania wymaga zasilania o napięciu +3.3v. Linię TXDA z GPS należy podłączyć do pinu RX na płytce Raspberry Pi, a RXDA (GPS) do TX (Raspberr Pi). Piny oznaczone NC nie wymagają żadnego podłączenia. Oprócz tego zalecane jest użycie antenty, którą należy podłączyć do pinu RF_IN. W projekcie jako antena, został wykorzystany przewód o długości około 10cm. Warto rozważyć podłączenie zasilania +3.3v do linii VBACKUP w celu zachowania informacji o ostatniej pozycji układu GPS w przypadku utraty zasilania. Po poprawnym podłączaniu w katalogu `/dev/` powinien pojawić się plik specjalny o nazwie `ttyAMA0` do którego zapisywane są informacje z GPS.



Rysunek 2: Schemat podłączeń



Rysunek 3: Wizualizacja połączeń

2.2 Implementacja programu

Do implementacji programu wykorzystano język C. W pierwszej kolejności oprogramowany został moduł wyświetlacza LCD. Wykorzystano bibliotekę udostępnioną dla komputerów Raspberry Pi - *WiringPi*, a z niej moduł LCD. Poniżej przedstawiona została funkcja inicjalizująca wyświetlacz.

```
int initLCD(){
    int lcd = lcdInit (2, 16,4, LCD_RS, LCD_E, LCD_D4, LCD_D5, LCD_D6, LCD_D7,0,0,0,0);
    if (lcd != 0){
        printf ("Nie udana inicjalizacja LCD! \n");
        return -1;
    }
    return lcd;
}
```

Wyświetlacz LCD korzysta ze sterownika HD44780, który można obsługiwać przy pomocy 4 lub 8 pinów do transferu danych. W przypadku korzystania z biblioteki `WiringPi` wymagane jest podłączenie tylko 4 pinów DB7, DB6, DB5 oraz DB4, które zostały podłączone kolejno do GPIO24, GPIO23, GPIO18, GPIO25. Oprócz tego LCD_Enable i LCD_RegisterSelect, odpowiedzialne za poprawną komunikację układu z Raspbery Pi. LCD zasilane jest napięciem +5v.

Na podstawie dokumentacji zaimplementowano funkcję do ekstrakcji danych z kodów *NMEA*. Funkcja ta parsuje ciągi znaków wczytane ze wejścia szeregowego, do którego podłączony jest moduł GPS. Następnie odczytane współrzędne wypisywanie są na LCD:

```
void parseNmea(char Buffer[255])
{
    if(Buffer[0] == '\$')
    {
    if (memcmp(Buffer+1,"GPGLL",5) == 0)
    {
```

```
            char *token = strtok(Buffer+6,",");
            char *lat;
            int i = 0;
            while(token != NULL)
            {
                switch(i)
                {
                //wypisanie szerokości geograficznej na ekran
                case 0:
                    printf(" Lat: %s ", token);
                    lcdPosition(lcd,0,0);
                    lcdPuts(lcd, "Lat: ");
                    lcdPuts(lcd, token);
                    break;
                 //wypisanie długości geograficznej na ekran
                case 2:
                    printf(" Lng: %s ", token);
                    lcdPosition(lcd,0,1); //drugi wiersz na ekranie LCD
                    lcdPuts(lcd, "Lng: ");
                    lcdPuts(lcd, token);
                    break;

                default:
                    printf("----------- BLAD!\n");
                    break;
                }
                i++;
                token = strtok(NULL,",");
                }
            printf("\n");
            }
        }
    }
```

Kolejnym fragmentem przygotowanego programu jest funkcja odczytująca dane z wejścia szeregowego, do którego podłączony został układ lokalizacji. Wykorzystano funkcję *serialOpen* z biblioteki *wiringSerial.h*. Odczyt danych przekazywanych przez GPS bazuje na czytaniu z pliku. Konieczne jest podanie ścieżki, oraz prędkości odczytu. Następnie następuje zapętlenie programu; kolejno pobierany jest odpowiedni kod *NMEA* i wywoływana wspomniana wcześniej metoda odpowiedzialna za parsowanie kodu i wyświetlanie danych na LCD.

```
void startTracking()
{
    char gps[65];
    int fd,flag=0; //uchwyt dla UART
    char arr[]="\$GPGLL";

    if((fd = serialOpen("/dev/ttyAMA0",9600)) < 0)
    {
        printf("%s\n", "Nie udalo sie podlaczyc do UART");
        return;
    }
    else
    {
        printf("Podlaczono UART\n");
    }
    int i = 0;
    char buffer[255];
    while(1)
    {
    int c;
        if(c=serialGetchar(fd))
```

```
            {
                if(c != '\n')
                {
                    buffer[i] = c;
                }
                else
                {
                    buffer[i] = '\0';
                    parseNmea(buffer);
                    i = -1;
                }
                i++;
            }
        }
    }
```

Funkcja *main* programu składa się z wywołania 4 funkcji opisanych powyżej.

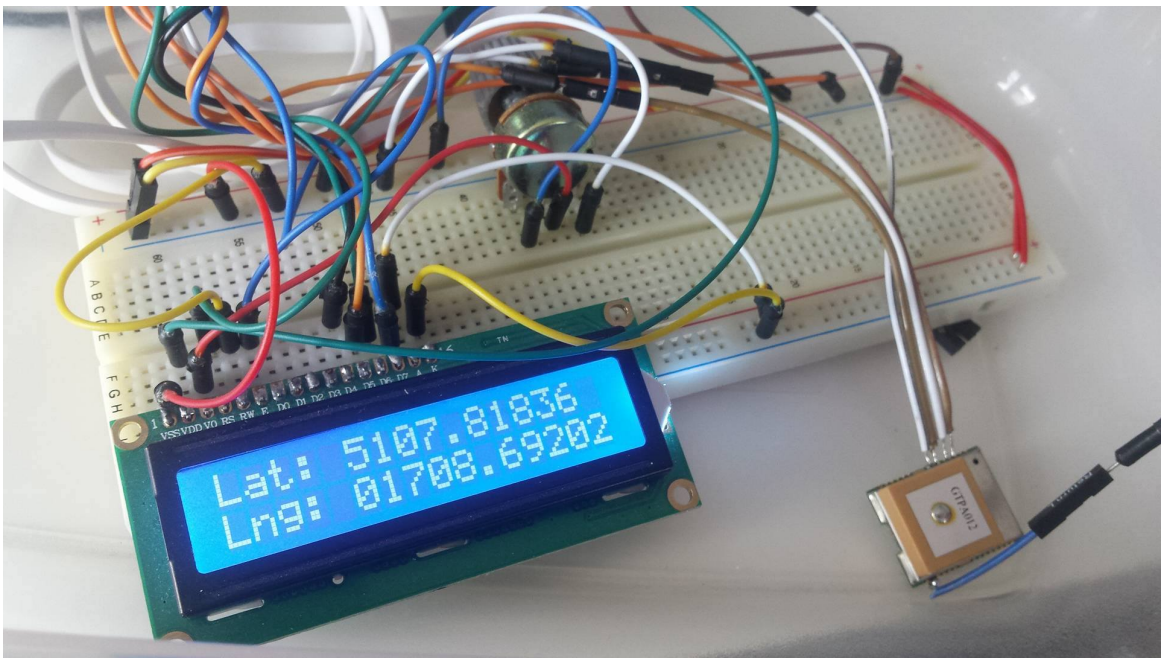
```
int main()
{
        wiringPiSetup(); // metoda inicjalizacyjna układ GPIO z biblioteki wiringPi.h
        lcd = initLCD(); // metoda inicjalizacyjna układ LCD
        startTracking(); // rozpoczęcie szukania lokalizacji
        getchar(); // oczekiwanie na znak
}
```

Cały projekty kompilowano poniższym poleceniem:

```
gcc lcd.c -o lcd -lwiringPi -lwiringPiDev
```

2.3 Testowanie

Początkowo testowano samo połączenie GPS z platformą Raspberry. Wykonanie komendy bashowej: *"cat /dev/ttyAMA0"* pozwala na wypisanie nieprzetworzonych danych na ekran konsoli. Niestety pierwsze próby zakończyły się fiaskiem. Powodem był brak anteny. Okazało się, że moduł lokalizacji wykorzystany w projekcie nie jest w nią zaopatrzony. Konieczne było dolutownaie przewodu o odpowiedniej długości, do odpowiedniego wyjścia modułu. Po tych czynnościach udało się otrzymać dane o położeniu. Program został skompilowany i uruchomiony na platformie Raspberry. Po włączeniu modułu konieczne jest odczekanie pewnej ilości czasu na ustablizowanie połączenia. Czas ten nie jest dłuższy niż 5 minut. Po tym czasie dane o lokalizacji pojawiają się na LCD:



Rysunek 4: Uzyskany efekt

3 Podsumowanie

Realizacja projektu pozwoliła na zapoznanie się z formatem kodów *NMEA*. Mnogość zagadnień teoretycznych obejmujących odczyt danych z portu szeregowego, wydobywanie danych z *NMEA* obsługę LCD jak również cześć praktyczna dotycząca lutowania i fizycznych połączeń była ciekawym i interesującym doświadczeniem. Najbardziej pouczające było szukanie przyczyn braku danych o lokalizacji, pomimo poprawnego podłączania modułu. W pierwszym momencie sądzono, że moduł został uszkodzony z powodu zbyt dużej temperatury podczas lutowania wyprowadzeń. Dołączenie anteny rozwiązało ten problem.

Literatura

[1] *https://github.com/WiringPi/WiringPi*,
dokumentacja oraz przykładowy kod źródłowy do obsługi LCD.

[2] *https://pinout.xyz*,
rozkład pinów na płytce Raspberry Pi 2.

[3] *https://botland.com.pl/index.php?controller=attachment&id_attachment=1334*,
dokumentacja modułu odbiornika GPS-GMS-U5LP.

[4] *http://home.mira.net/ gnb/gps/nmea.html*,
informacje dodatkowe na temat formatu NMEA 0183.

[5] *http://www.cplusplus.com/doc/*,
dokumentacja języka C++.

[6] *http://elinux.org/RPi_Serial_Connection*,
informacje na temat połączenia UART.

[7] *http://wiringpi.com/reference/serial-library/*,
informacje o bibliotece wiringSerial.