Politechnika Łódzka
Instytut Informatyki

PRACA DYPLOMOWA INŻYNIERSKA

Stacja pogodowa ze zdalnym dostępem zbudowana w oparciu o telefon komórkowy

Wydział Fizyki Technicznej, Informatyki i Matematyki Stosowanej
Promotor: dr inż. Michał Morawski
Dyplomant: Piotr Marcińczyk
Nr albumu: 180633
Kierunek: informatyka
Specjalność: inżynieria oprogramowania i analiza danych

Łódź, 22 marca 2016

Instytut Informatyki
90-924 Łódź, ul. Wólczańska 215, budynek B9
tel. (42) 631-27-97, (42) 632-97-57, fax (42) 630-34-14, e-mail: office@ics.p.lodz.pl

Spis treści

Spis skrótów

3GPP	3rd Generation Partnership Project
AHB	Advanced High-performance Bus
APB	Advanced Peripheral Bus
ARM	Advanced RISC Machine
ASCII	American Standard Code for Information Interchange
BCD	Binary-Coded Decimal
CPU	Central Processing Unit
CTS	Clear To Send
DMA	Direct Memory Access
GMT	Greenwich Mean Time
GSM	Global System for Mobile Communications
IAP	In Application Programming
IDE	Integrated Development Environment
IRA	International Reference Alphabet
ITU	International Telecommunication Union
ITU-T	Sektor Normalizacji Telekomunikacji ITU
I²C	Inter-Integrated Circuit
JTAG	Joint Test Action Group
LDO	Low-Dropout Regulator
MCO	Microcontroller Clock Output
MCU	Microcontroller unit
OTG	USB On-The-Go
PCB	Printed Circuit Board
PDU	Protocol Data Unit
PLL	Phase-Locked Loop
RAM	Random-Access Memory
RCC	Reset and clock control
RTC	Real Time Clock
RTOS	Real-time operating system
RTS	Ready To Send
SMS	Short Message Service
SPI	Serial Peripheral Interface
SWD	Serial Wire Debug
THT	Trough-Hole Technology
UKE	Urząd Komunikacji Elektronicznej
URC	Unsolicited Result Code
USART	Universal Synchronous and Asynchronous Receiver and Transmitter
USB	Universal Serial Bus
VCO	Voltage-Controlled Oscillator

Streszczenie

Niniejsza praca inżynierska opisuje sposób realizacji przykładowej stacji pogodowej, komunikującej się z użytkownikiem przy wykorzystaniu sieci telefonii komórkowej. System wbudowany opiera się na mikrokontrolerze **STMicroelectronics STM32F401RE**, umieszczonym na płytce uruchomieniowej **NUCLEO-F401RE**. Oprogramowanie układu zostało stworzone w języku C, przy wykorzystaniu środowiska **Ac6 System Workbench for STM32**. Za system operacyjny czasy rzeczywistego posłużył **FreeRTOS**. Pomiary są realizowane za pomocą dwóch czujników: wilgotnościomierz **Aosong AM2302** oraz ciśnieniomierz **Bosch BMP180**, obydwa posiadające wbudowany termometr. Dane przekazywane są krótkimi wiadomościami tekstowymi SMS, wysyłanymi z telefonu **Sony Ericsson K300i**, połączonego ze stacją za pomocą zmodyfikowanego kabla serwisowego **DCU-11**. Komunikacja odbywa się asynchronicznym protokołem szeregowym, przy wykorzystaniu komend AT Hayesa. Wskazania stacji mogą być uzyskane na żądanie lub cyklicznie według ustawionego interwału. Konfiguracja parametrów odbywa się za pomocą terminala tekstowego, przez wirtualny port szeregowy, po podłączeniu kabla USB. Uzyskany system może służyć do monitorowania warunków w szklarni lub ogrodzie. Budowa stacji pozwala na jej łatwe rozbudowanie, dodanie nowych funkcjonalności lub wykorzystanie zaimplementowanego sposobu komunikacji w innym systemie.

Abstract

The following engineering thesis describes realisation of a weather monitoring station, communicating with the user over a cellular network. The embedded system uses **STMicroelectronics STM32F401RE** microcontroller, placed on **NUCLEO-F401RE** development board. The software has been created in C programming language, using **Ac6 System Workbench for STM32** integrated environment. **FreeRTOS** serves as a real-time operating system. Measurements are done by two sensors: **Aosong AM2302** hygrometer and **Bosch BMP180** barometer, both having built-in termometers. Data is transferred in SMS messages, sent from **Sony Ericsson K300i** mobile phone, connected to the station with a modified **DCU-11** service cable. Communication is based on an asynchronous serial protocol using Hayes AT command set. The values measured by the station can be retrieved on demand or cyclically after configured interval. Settings can be changed using a text terminal connected to the virtual serial port over USB cable. Created system could be used for monitoring meteorological conditions of a greenhouse or a garden. Construction of the station allows for its simple extending, adding new functionalities or using implemented way of communication in another systems.

Rozdział 1

Wprowadzenie

1.1. Wstęp

Szybko postępujący rozwój techniki powoduje, że urządzenia elektroniczne posiadają coraz większe możliwości techniczne i funkcjonalne, zarazem mieszcząc się w mniejszych rozmiarach. Jest to jeden z powodów, dla którego wiele osób posiada własny telefon komórkowy. Często są to wysoko zaawansowane urządzenia, z parametrami dorównującymi komputerom osobistym. Między innymi dlatego, łączność mobilna jest wybierana jako sposób komunikacji ze zdalnymi przyrządami. Przykładem takich urządzeń są m.in. systemy alarmowe, punkty pomiaru natężenia ruchu drogowego, czy stacje pogodowe. W celu umożliwienia połączenia ze stacjami bazowymi sieci komórkowej niezbędny jest modem **GSM**. Większość z nich umożliwia wysyłanie krótkich wiadomości tekstowych **SMS**, wykonywanie połączeń głosowych, a także wymianę danych z siecią internet. Ceny nowych modemów utrzymują się na wysokim poziomie, co może zniechęcać do wykorzystania tej technologii w projekcie. Można jednak obniżyć koszt całego systemu, wykorzystując telefon komórkowy starszej generacji, który nadal działa, lecz nie jest już używany – na przykład z powodu niewystarczającej funkcjonalności dla użytkownika. Takie telefony nietrudno znaleźć na rynku wtórnym i niedużym kosztem uzyskać możliwość zdalnej komunikacji tworzonego systemu.

Ponadto do wykorzystania telefonu nie są wymagane jego wszystkie funkcjonal-

ności. Dlatego urządzenia z niesprawnymi wyświetlaczami, głośnikami lub klawiaturami także są użyteczne w tego typu projektach. Komunikacja odbywa się przeważnie za pomocą kabla, podczerwieni lub technologii Bluetooth z wbudowanym w telefon modemem. Dzięki temu jeszcze bardziej zwiększa się dostępność potencjalnego urządzenia, które można zastosować jako odpowiednik drogiego modemu bezprzewodowej sieci telefonicznej.

1.2. Cele i zakres pracy

Celem niniejszej pracy jest stworzenie prostej stacji pogodowej, która będzie przykładem systemu wykorzystującego sposób zdalnej komunikacji opisany we wstępie. Stacja będzie badać podstawowe parametry atmosferyczne tj. temperaturę, wilgotność względną oraz ciśnienie powietrza. Następnie dane powinny zostać przesłane w formie krótkiej wiadomości tekstowej (**SMS**) do innego urządzenia, najczęściej telefonu. Z tego wynika, że połączenie odbywa się za pomocą bezprzewodowej sieci telefonii komórkowej. Takie rozwiązanie zapewnia łączność z systemem wszędzie tam, gdzie jest zasięg **GSM**, czyli praktycznie w każdym miejscu Polski, a także na świecie[1]. Jak wspomniano we wstępie, atrakcyjnym rozwiązaniem jest wykorzystanie telefonu starszej generacji jako urządzenia posiadającego niezbędny modem, który umożliwi tego typu komunikację.

Głównym elementem sterującym w zaproponowanym systemie będzie mikrokontroler, zbierający dane z czujników pogodowych oraz komunikujący się z podłączonym telefonem komórkowym przez łącze szeregowe. Będzie on także zajmował się przetwarzaniem przychodzących poleceń i powinien zapewniać wykrywanie pojawiających się błędów oraz informować o nich użytkownika. Oprogramowanie dla mikrokontrolera powinno być w miarę możliwości uniwersalne pod względem funkcjonalności, tzn. umożliwiać w łatwy sposób podłączenie do innego systemu niż stacja pogodowa (np. systemu alarmowego lub zestawu przekaźników).

Praca zawiera również elementy teoretyczne, potrzebne do konstruowania takie-

[1] Na podstawie map zasięgu UKE `http://btsearch.pl/` oraz Mobile World Live `http://maps.mobileworldlive.com/network.php?cid=30&cname=Poland` (dostęp: 16.02.2016)

go systemu, a także analizę skuteczności wybranych rozwiązań, w tym odporność na różnego rodzaju błędy i sytuacje wyjątkowe. Dodatkowo zaproponowano możliwe alternatywne zastosowania systemu i dalsze sposoby rozwoju układu. Warto także podkreślić pozytywny wpływ na środowisko poprzez wykorzystanie urządzeń, które w przeciwnym wypadku zostałyby przeznaczone do utylizacji.

W rozdziale 2 dokonano porównania sprzętu i oprogramowania potrzebnego do wykonania systemu tego typu oraz umotywowano wybór narzędzi do tej pracy. Rozdział 3 zawiera szczegółowy opis realizacji praktycznej części systemu. W kolejnym rozdziale 4 znajduje się podstawowy podręcznik użytkownika, czyli instrukcja obsługi zrealizowanego systemu. W końcowej części pracy zawarto skuteczność zastosowanego rozwiązania i obserwacje związane ze sprawnością układu, wraz z oceną skuteczności reakcji na błędy.

Rozdział 2

Opis sprzętu i narzędzi

2.1. Mikrokontroler

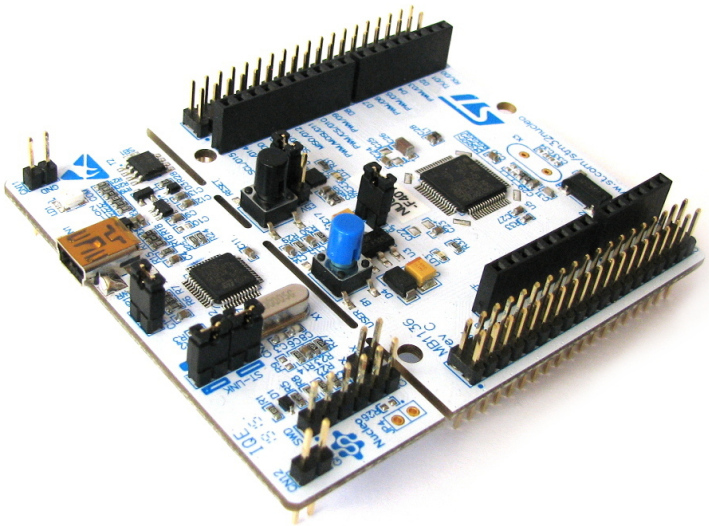
Szybki rozwój technologii produkcji oraz ich szerokie zastosowanie, a co za tym idzie – duży popyt – spowodował, że 32-bitowe procesory ARM stanowią bardzo atrakcyjną alternatywę dla urządzeń 8-bitowych. Ich ceny są porównywalne, a możliwości tych pierwszych znacznie bardziej rozbudowane. Ponadto zastosowanie takiego mikrokontrolera (MCU) pozwoli na łatwiejszą ewentualną rozbudowę projektu, a także zastosowanie systemu operacyjnego czasu rzeczywistego – RTOS. Tego typu rozwiązanie w łatwy sposób umożliwia zarządzanie zadaniami przydzielonymi systemowi, sprawne dodawanie nowych oraz określanie ich priorytetów.

Do systemu tworzonego w pracy został wybrany kontroler firmy **STMicroelectronics**, a konkretnie model **STM32F401RE**. Jest to procesor z grupy ARM Cortex-M4, posiadający poniżej przedstawione parametry [1]:

- taktowanie CPU do 84 MHz
- 512 kB pamięci trwałej FLASH
- 96 kB pamięci ulotnej Static RAM
- zasilanie napięciem od 1,7 do 3,6 V
- 11 liczników + timer systemowy
- interfejsy komunikacyjne: USART, I²C, SPI
- USB 2.0 full-speed z OTG

- zegar czasu rzeczywistego RTC podtrzymywany baterią
- 16 kanałów kontrolera DMA
- interfejsy debugowania: SWD oraz JTAG.

Układ produkowany jest tylko w obudowach do montażu powierzchniowego, co powoduje utrudnione lutowanie bez specjalistycznego sprzętu. Dlatego w prototypie systemu można wykorzystać gotową płytkę uruchomieniową oferowaną przez producenta układu. Do najpopularniejszych zestawów należą **DISCOVERY** oraz **NUCLEO**. Pierwszy z nich charakteryzuje się dużą ilością elementów takich jak przyciski, diody LED, akcelerometr, żyroskop, wyświetlacz, mikrofon, czy wzmacniacz audio. Drugi natomiast jest bardziej uniwersalny – na płytce znajdują się tylko elementy niezbędne do uruchomienia zestawu, w tym programator i wyjścia na własne podzespoły oraz rozszerzenia, także kompatybilne z platformą **Arduino**. Ponieważ większość elementów znajdujących się w zestawach **DISCOVERY** nie zostałaby wykorzystana w projektowanym systemie, ostatecznie wybrano płytkę startową **NUCLEO-F401RE**, widoczna na ilustracji 2.1.



Ilustr. 2.1: Wygląd płytki startowej NUCLEO-F401RE

Zestaw NUCLEO posiada przyłączony programator **ST-LINK/V2**, który może zostać odłączony przez odcięcie PCB w wyróżnionym miejscu [3]. Dzięki temu łatwo wykorzystać go jako uniwersalny programator do przyszłych projektów. Pro-

gramator posiada wlutowany rezonator kwarcowy 8 MHz, będący również podłączony do wyjść głównego MCU. Do podłączenia komputera, a także umożliwienia zasilania, zostało wyprowadzone gniazdo mini-USB. Programator posiada wbudowany konwerter UART↔USB, który umożliwia jednoczesną pracę debuggera oraz komunikację szeregową z układem. Standardowo linie komunikacyjne konwertera są wewnętrznie podłączone na płytce, dzięki czemu nie ma konieczności ręcznego ingerowania w tym celu.

Na płytce NUCLEO znajdują się dwa przyciski: czarny i niebieski, oznaczone odpowiednio **RESET** i **USER**. Zadaniem pierwszego jest reset głównego mikrokontrolera, umożliwiający ponowne uruchomienie programu znajdującego się w pamięci. Drugi jest podłączony do wyjścia `PC13` kontrolera i może zostać zaprogramowany przez użytkownika.

Po obydwu stronach znajdują się wyprowadzenia z mikrokontrolera w postaci listwy goldpinów, umożliwiające podłączenie do niego dowolnych podzespołów. Dostęp do nich jest możliwy zarówno od góry, jak i od dołu.

2.2. Środowisko programistyczne

Na rynku można znaleźć wiele zintegrowanych środowisk programistycznych (IDE), pozwalających na tworzenie oprogramowania dla procesorów ARM. Spora część z nich jest płatna lub posiada darmowe wersje ograniczone czasowo i maksymalnym rozmiarem kodu wynikowego. Przykładami takich środowisk są:

- **IAR Embedded Workbench for ARM**[1];
- **Keil MDK-ARM**[2];

Ich podstawową wadą jest brak możliwości kompilacji kodu na platformę PC. To z kolei uniemożliwia testowanie własnych funkcji pomocniczych, które są niezależne od architektury i mogą zostać uruchomione lokalnie, bez udziału mikrokontrolera.

[1] IAR Embedded Workbench for ARM, `https://www.iar.com/iar-embedded-workbench/` (dostęp 17.02.2016)
[2] Keil MDK-ARM, `http://www.keil.com/arm/mdk.asp` (dostęp 17.02.2016)

Alternatywą dla komercyjnych rozwiązań są narzędzia posiadające otwarty kod źródłowy, a przez to w pełni darmowe. Większość z nich jest na tyle uniwersalna, że wymaga dużej ilości konfiguracji, w celu uzyskania całkowicie sprawnego środowiska. To może być pewną przeszkodą, ponieważ wymaga sporej ilości czasu, który należy poświęcić przed samym rozpoczęciem procesu tworzenia oprogramowania. Istnieją także rozwiązania już skonfigurowane pod konkretną platformę, jak na przykład wybrany w tej pracy **Ac6 System Workbench for STM32** udostępniony na platformie **OpenSTM32 Community**[3]. Jest on oparty na środowisku **Eclipse**[4], co umożliwia uruchomienie go praktycznie na każdym popularnym systemie operacyjnym. Ponadto zapewnia dużą ilość wtyczek tworzonych przez społeczność, pozwalających na dostosowanie środowiska do potrzeb programisty. Do kompilacji wykorzystywane są narzędzia **GNU ARM Toolchain** na architekturę `arm-none-eabi`. Językiem programowania wykorzystanym w pracy jest **C** w standardzie `gnu99`.

Ponadto do komunikacji z programatorem wykorzystane zostało oprogramowanie **OpenOCD**. Umożliwia ono nie tylko przesyłanie programów, ale także uruchomienie programu w trybie debugowania tzn. przetwarzania programu instrukcja po instrukcji, z rzeczywistym podglądem wszystkich zmiennych i rejestrów procesora, a także możliwością ich ręcznej modyfikacji. To z kolei pozwala na znacznie łatwiejsze wyszukiwanie błędów oraz sprawdzanie poprawności wykonywania zaprogramowanych funkcji.

2.3. System operacyjny czasu rzeczywistego

W niewielkich projektach zastosowanie podejścia nieskończonej pętli w głównej funkcji programu bywa wystarczające. Tego typu rozwiązanie polega na ciągłym sprawdzaniu stanu podzespołów i wykonywaniu adekwatnych do niego działań. Dodatkowo można wykorzystać przerwania, które wywoływane bezpośrednio

[3] OpenSTM32 Community, *System Workbench for STM32*, `http://www.openstm32.org/System+Workbench+for+STM32` (dostęp 17.02.2016)

[4] *The Eclipse Foundation open source community website*, `https://eclipse.org/` (dostęp 17.02.2016)

po zmianie stanu, pozwalają na sprawniejsze spełnianie limitów czasowych wymaganych przez system.

Problem jednak pojawia się wraz ze wzrostem funkcjonalności wymaganych od systemu wbudowanego. Pętla główna programu staje się mniej czytelna, a dodawanie nowych zadań powoduje tworzenie się kodu coraz trudniejszego w zarządzaniu. Mała przewidywalność kolejności działań powstającego programu zwiększa ryzyko popełniania błędów. Z pomocą przychodzi system operacyjny czasu rzeczywistego, który zajmuje się m.in. przełączaniem kontekstu zadań, zachowując wymagania dotyczące spełniania limitów czasowych. Istnieje niezliczona ilość takich systemów, w tym rozwiązania komercyjne. Do takich należą na przykład: **BeRTOS**[5], **DDC-I HeartOS**[6], **FreeRTOS**[7], **SafeRTOS**[8], **OpenRTOS**. Te dwa ostatnie są płatną odnogą systemu **FreeRTOS**, opracowaną przez firmę **WITTENSTEIN** i zapewniającą certyfikaty bezawaryjności systemu.

Do pracy wybrano **FreeRTOS V8.2.3**, ze względu na dużą popularność, przejrzystą dokumentację oraz stały rozwój systemu – a przez to niskie prawdopodobieństwo błędnego działania. Umożliwia on stworzenie osobnych zadań do każdej funkcjonalności i ustawienie im priorytetów określających ważność. Przełączenie do zadania o niższym priorytecie nastąpi dopiero gdy to ważniejsze jest uśpione lub zablokowane (np. na semaforze). **FreeRTOS** udostępnia wiele opcji konfiguracji dostępnych w nagłówku `FreeRTOSConfig.h`. Do najważniejszych należą:

- `USE_PREEMPTION` – automatyczne wywłaszczanie kontekstu procesora;
- `TICK_RATE_HZ` – częstotliwość zegara jądra systemu z jaką działa przerwanie, w którym następuje ewentualne wywłaszczenie;
- `LIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY` – najwyższy priorytet przerwania, który może odwoływać się do funkcji systemowych – przerwania o wyższym priorytecie będą niezależne od systemu;

Do działania, jądro systemu wymaga, oprócz nagłówków, tylko kilku plików

[5] `http://www.bertos.org/`
[6] `http://www.ddci.com/products_heartos.php`
[7] `http://www.freertos.org/`
[8] `http://www.highintegritysystems.com/safertos/`

z kodem źródłowym:

- `tasks.c` – zarządzanie zadaniami i przydzielaniem kontekstu procesora;
- `list.c` – implementacja list (wykorzystywanych m.in. przez schedulera);
- `queue.c` – implementacja synchronizowanych kolejek;
- `port.c` i `portmacro.h` – wszelkie definicje zależne od architektury procesora i kompilatora;
- `MemMang/heap_x.c` – jeden z 5 dostępnych menadżerów sterty, zapewniający przydzielanie i zwalnianie pamięci.

2.4. Warstwa abstrakcji sprzętowej

Do zarządzania podzespołami mikrokontrolera została wykorzystana biblioteka **Standard Peripheral Library V1.5.1** (w skrócie oznaczana **StdPeriph**), dostarczona przez **STMicroelectronics** (różnice pomiędzy poszczególnymi rodzinami procesorów są niewielkie, dlatego dokumentacja [4] odnosi się do **STM32F3xxx**). Dostarcza ona mapę rejestrów procesora wraz z konfiguracjami na poziomie bitów oraz także zbiór funkcji zapewniających poziom abstrakcji od sprzętu, dzięki czemu istnieje możliwość manipulacji podzespołami bez konieczności znajomości konkretnych rejestrów. To z kolei zwiększa czytelność kodu, ułatwia dokonywanie zmian oraz umożliwia przenoszenie programu między różnymi modelami procesora tego producenta, minimalizując ryzyko błędów związanych różnicami w architekturze.

2.5. Czujniki pogodowe

Czujniki wykorzystane do pomiaru warunków meteorologicznych to **Aosong AM2302** oraz **Bosch BMP180**. Pierwszy z nich to cyfrowy higrometr pojemnościowy, służący do pomiaru wilgotności względnej powietrza. Widoczny jest na ilustracji 2.2. Posiada wbudowany termometr służący do kompensacji temperaturowej pomiarów, którego wskazania również mogą zostać odczytane. Należy on do rodziny czujników **DHT-22** i jest znany także pod nazwą **RHT03**.

Ilustr. 2.2: Czujnik wilgotności Aosong AM2302

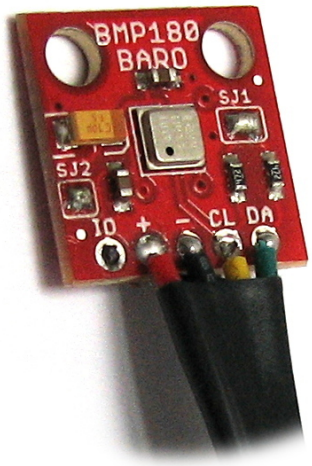
Zakres działania czujnika odczytany z dokumentacji [5] został podany w tabeli 2.1. Układ zasilany jest napięciem od 3,3 do 5 V, co pozwala na bezpośrednie przyłączenie go do układu.

Parametr	**Zakres**	**Dokładność**	**Rozdzielczość**
wilgotność, %	od **0,0** do **99,9**	±2	±0,1
temperatura, °C	od **-40** do **80**	±0,5	±0,1

Tab. 2.1: Parametry czujnika wilgotności AM2302

Komunikacja czujnika **AM2302** z mikrokontrolerem realizowana jest za pomocą jednej linii danych. Protokół częściowo przypomina interfejs **Maxim-Dallas 1-Wire** [7], jednak nie uwzględnia możliwości adresowania przyłączonych urządzeń. Z tego powodu każde urządzenie tego typu wymaga osobnej linii komunikacyjnej. Ponadto poszczególne czasy wymagane przez protokół różnią się w pewnym stopniu między dokumentacjami [5, 6] czujników z rodziny **DHT-22**, dlatego w pracy uwzględniono kilkuprocentowy margines błędu (uwzględniający również niedokładności oscylatorów poszczególnych układów).

Drugi z czujników – **Bosch BMP180** – piezorezystancyjny miernik ciśnienia, jest umieszczony na płytce **SparkFun**, posiadającej wyprowadzenia do montażu przewlekanego (THT). Fabrycznie produkowany jest tylko w niewielkich obudowach *Land Grid Array*, trudnych do lutowania bez specjalnego sprzętu – widoczny w środkowej części płytki na ilustracji 2.3.



Ilustr. 2.3: Czujnik ciśnienia BMP180 na płytce SparkFun

Wskazania czujnika znajdują się w zakresach przedstawionych w tabeli 2.2 według dokumentacji [8]. Komunikacja z układem odbywa się za pomocą standardowego protokołu I²C przy częstotliwości linii zegarowej do 400 MHz.

Parametr	Zakres	Dokładność	Rozdzielczość
ciśnienie, hPa	od **300** do **1100**	±0,12	±0,01
temperatura, °C	od **-40** do **85**	±0,5	±0,1

Tab. 2.2: Parametry czujnika ciśnienia BMP180

2.6. Telefon do komunikacji

Wybierając telefon umożliwiający zdalne połączenie z systemem, należy zadbać o dobrze udokumentowany protokół, który jest wykorzystywany do komunikacji oraz łatwy, mało inwazyjny sposób podłączenia. Producenci często opracowują własne sposoby komunikacji i nie udostępniają niezbędnych specyfikacji, co uniemożliwia sprawne połączenie z telefonem. Przykładami wykorzystywanych protokołów są **MBUS** i **FBUS** zaprojektowane przez firmę **Nokia**. Opierają się one na asynchronicznej transmisji szeregowej danych. Różnią się między sobą głównie prędkością połączenia oraz kierunkowością – odpowiednio 9600 bps półdupleks i 115200 bps pełny dupleks. Każdy przesyłany komunikat składa się z ramki w formacie binar-

nym. Więcej informacji na ten temat można znaleźć na stronie[9] otwartego programu **Gnokii**, który umożliwia komunikację telefonu z komputerem przy wykorzystaniu tych protokołów i zawiera ich szczegółową specyfikację.

Znacznie bardziej popularnym standardem jest zbiór komend **AT Hayesa** (od *attention*) opracowanym przez Dennisa Hayesa na potrzeby analogowych modemów telefonicznych[10]. Jest on ciągle rozwijany oraz powszechnie wykorzystywany przez wiele modemów (również GSM). Komunikacja jest w znacznej części tekstowa, a jej realizacja umożliwia przesyłanie poleceń zarówno maszynowo, jak i ręcznie przez człowieka. To z kolei pozwala na łatwą analizę przebiegu transmisji oraz sprawne wykrywanie błędów. Pierwotnie standard został zatwierdzony przez Sektor Normalizacji Telekomunikacji **ITU-T** [9], a współcześnie specyfikację oraz wprowadzanie nowych komend zapewnia grupa **3GPP** [10]. Do najpopularniejszych telefonów obsługujących ten protokół należą niektóre modele firmy Nokia (np. 5140, 6020, 6310, 6510), a także **Sony Ericsson** (produkowane przed 2005 rokiem umożliwiają łączność zgodną z UART, podczas gdy nowsze wymagają implementacji obsługi interfejsu USB). Dokładna lista kompatybilnych urządzeń znajduje się w specyfikacji [11].

System wbudowany zrealizowany w ramach pracy wykorzystuje telefon Sony Ericsson K300i, widoczny na ilustracji 2.4. Komunikacja oraz podłączenie źródła zasilania jest możliwe za pomocą portu znajdującego się w dolnej części urządzenia. Takie rozwiązanie pozwala na uzyskanie łączności, bez dodatkowej ingerencji w telefon. Dodatkowo dla zapewnienia lepszej kompatybilności z różnymi urządzeniami, które mogą zostać podłączone do stacji, wykonano testy wykorzystując drugi aparat Sony Ericsson K700i przedstawiony na ilustracji 2.5.

[9] Gnokii project, `https://gnokii.org/` (dostęp 19.02.2016)

[10] History of computers: *Modem*, `http://history-computer.com/ModernComputer/Basis/modem.html` (dostęp 19.02.2016)

Ilustr. 2.4: Główny telefon – Sony Ericsson K300i

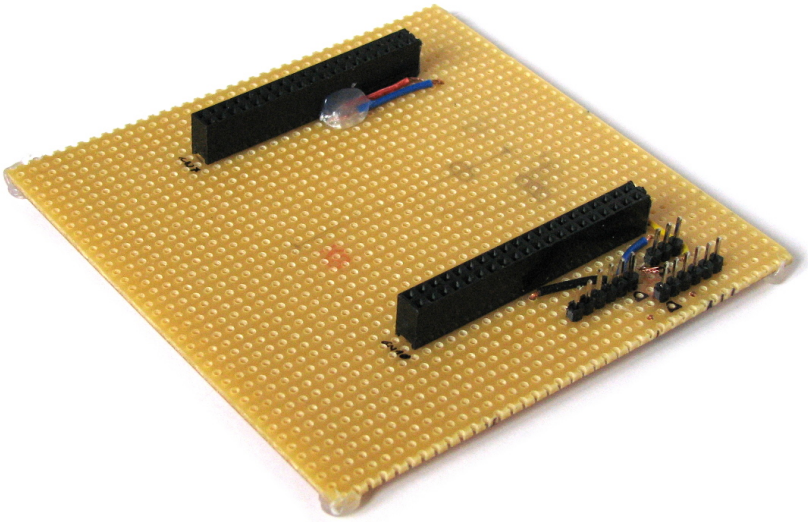
Ilustr. 2.5: Dodatkowy telefon – Sony Ericsson K700i

Rozdział 3

Realizacja praktyczna

3.1. Płytka rozszerzająca

Na potrzeby pracy, została stworzona płytka rozszerzająca przedstawiona na ilustracji 3.1. Podłączana jest do dolnej części wyprowadzeń NUCLEO przez dwie podwójne żeńskie listwy goldpin. Znajdują się na niej trzy wyjścia: na czujniki pogodowe oraz telefon wykorzystywany do zdalnej komunikacji. Takie rozwiązanie pozwala na wygodną wymianę poszczególnych modułów, a także zapewnia miejsce na dalsze rozszerzanie systemu bez ingerencji w oryginalną płytkę.



Ilustr. 3.1: Własna płytka rozszerzająca z gniazdami na NUCLEO

3.2. Inicjalizacja podzespołów

Do taktowania układu zastosowano sygnał oscylatora **MCO (Microcontroller Clock Output)** wychodzący z programatora **ST-LINK** znajdującego się na płytce NUCLEO. Jego częstotliwość wynosi $f_{\text{in}} = 8$ MHz i pierwotnie pochodzi z rezonatora kwarcowego `X1` [3]. Wykorzystując wbudowaną w mikrokontroler pętlę PLL, ustawiono taktowanie układu na częstotliwość $f_{\text{out}} = 64$ MHz. Konfiguracja wymaganych do tego rejestrów znajduje się w głównym pliku źródłowym biblioteki **StdPeriph** (opisanej w rozdziale 2.4) o nazwie `system_stm32f4xx.c`. Zależności z instrukcji [2] zostały zaprezentowane na równaniach (3.1) i (3.2):

$$f_{\text{VCO}} = f_{\text{in}} \cdot \frac{\texttt{PLL\_N}}{\texttt{PLL\_M}} \tag{3.1}$$

$$f_{\text{out}} = \frac{f_{\text{VCO}}}{\texttt{PLL\_P}} = 64 \text{ MHz} \tag{3.2}$$

Dodatkowo należy spełnić wymagania dotyczące przedziałów konkretnych częstotliwości oraz parametrów, które zostały podane we wzorach od (3.3) do (3.7):

$$192 \text{ MHz} \leqslant f_{\text{VCO}} \leqslant 432 \text{ MHz} \tag{3.3}$$

$$1 \text{ MHz} \leqslant \frac{f_{\text{in}}}{\texttt{PLL\_M}} \leqslant 2 \text{ MHz} \tag{3.4}$$

$$192 \leqslant \texttt{PLL\_N} \leqslant 432 \tag{3.5}$$

$$2 \leqslant \texttt{PLL\_M} \leqslant 63 \tag{3.6}$$

$$\texttt{PLL\_P} \in \{2; 4; 6; 8\} \tag{3.7}$$

Zalecana wartość częstotliwości w (3.4) wynosi 2 MHz, co zapewnia stabilniejszą pracę pętli PLL i zarazem całego układu. Z powyższych założeń można określić konfigurację najlepszych parametrów, która została przedstawiona w tabeli 3.1.

Parametr	Wartość
PLL_M	4
PLL_N	192
PLL_P	6

Tab. 3.1: Ustawione wartości rejestrów PLL

W mikrokontrolerze **STM32F401** wyróżnia się kilka głównych magistrali: **AHB**, **APB1** (mała prędkość) i **APB2** (duża prędkość). Większość podzespołów komunikuje się między sobą jedną z tych trzech linii. Zaraz po uruchomieniu mikrokontrolera, w celu oszczędzania energii, peryferia są odłączone od magistrali. Z tego powodu, należy programowo umożliwić działanie wybranych podzespołów przestawiając rejestry z grupy **RCC** (Reset and clock control). W zależności od linii wybranego modułu, służy do tego funkcja `RCC_AxBnPeriphClockCmd()`, gdzie `x` jest identyfikatorem `H` lub `P`, a `n` numerem od `1` do `3`.

3.3. Komunikacja z czujnikami

Obydwa sensory pogodowe zasilane są napięciem 3,3 V pochodzącym ze stabilizatora liniowego o niskim spadku napięcia (**LDO**) znajdującego się na płytce NUCLEO i oznaczonego `U4` [3]. Wyjście regulatora jest wyróżnione na schemacie jako `+3V3`.

3.3.1. Czujnik wilgotności AM2302

Czujnik posiada jedną linię komunikacyjną, domyślnie podciąganą do napięcia zasilania. Aby zainicjować połączenie, mikrokontroler obniża napięcie do masy, na czas co najmniej 1 ms, następnie z powrotem podciąga do zasilania. Następnie po czasie T_{go}, czujnik wykonuje sekwencję odpowiedzi poprzez wprowadzenie stanu niskiego przez T_{rel} i wysokiego T_{reh}. Dalej następuje przesłanie 5 oktetów danych, w których znajdują się (najbardziej znaczący bit i bajt jako pierwsze): dwa reprezentujące wilgotność, dwa temperaturę i jeden na sumę kontrolną, czyli 8 najmłodszych bitów sumy pierwszych 4 bajtów danych. Poszczególne bity różnią się czasem stanu wysokiego $T_{\text{H}x}$, podczas gdy stan niski trwa T_{L}. Komunikacja kończy się ściągnięciem do stanu niskiego przez okres T_{end}. Przedziały dopuszczalnych czasów w mikrosekundach (µs), według [5, 6] z uwzględnieniem marginesu błędu oscylatorów, zostały zebrane w tabeli 3.2.

Podzespoły znajdujące się w mikrokontrolerze nie oferują bezpośrednio sprzętowej obsługi protokołu czujnika. Istnieje możliwość wykorzystania modułu USART

Czas	Min	Max
T_{go}	0	200
T_{rel}	70	90
T_{reh}	70	90
T_{L}	40	70
T_{H0}	20	45
T_{H1}	50	80
T_{end}	40	60

Tab. 3.2: Przedziały czasów w µs dla protokołu czujnika AM2302

do komunikacji w standardzie 1-Wire[1], jednak programowa implementacja przy użyciu pinu **GPIO** jest w tym wypadku wystarczająca. Linia komunikacyjna czujnika została podłączona do pinu `PC9` mikrokontrolera. Skonfigurowano go w kierunku wejściowym z wewnętrznym rezystorem podciągającym (*pull-up*) oraz ze stanem niskim w trybie wyjścia, w taki sposób, aby zmiana kierunku była wystarczająca do osiągnięcia odpowiednich poziomów sygnału. Pomiar czasu wymaganego przez poszczególne etapy protokołu zrealizowano wykorzystując 16-bitowy licznik `TIM10` z dzielnikiem ustawionym na wartość $f_{\text{out}}/1000000 = 64$. To pozwala uzyskać cykl zegara wynoszący w przybliżeniu 1 µs.

Otrzymane dane z sensora są w postaci stałoprzecinkowej z zapisem znak-moduł. Aby otrzymać wartość wilgotności lub temperatury, należy podzielić ją przez 10. Jednak w celu przyspieszenia działań, postanowiono przechowywać oryginalne wskazania, a do przekształcania ich na postać tekstową, zaimplementować własną procedurę o nazwie `printx()`. Przyjmuje ona wskaźnik do funkcji wypisującej pojedynczy znak (np. do transmitera USART lub bufora tekstowego), co czyni ją bardziej uniwersalną od standardowego `printf()`. Za wypisywanie liczb stałoprzecinkowych odpowiada flaga „`%.#b`”, gdzie „#” określa ilość miejsc po przecinku.

3.3.2. Czujnik ciśnienia BMP180

Odczyt z sensora wykonywany jest za pomocą modułu I²C. Linie komunikacyjne zostały podłączone pod piny oznaczone `I2C1`: `PB8` (SCL) i `PB9` (SDA), a następnie

[1] Maxim Integrated: *Using a UART to Implement a 1-Wire Bus Master*, `https://www.maximintegrated.com/en/app-notes/index.mvp/id/214` (dostęp 22.02.2016)

skonfigurowane w tryb funkcji alternatywnej AF (*Alternate Function*). Taktowanie łącza ustawiono na 100 kHz, aby zmniejszyć podatność na błędy komunikacji. Adres czujnika wynosi `0xEE` (dla odczytu `0xEF`) [8]. Sensory tego typu posiadają wewnętrzną pamięć **EEPROM** o różnej zawartości, na której znajdują się unikalne dane kalibracyjne, niezbędne do wyznaczenia dokładnej wartości pomiaru. Są one odczytywane zaraz po uruchomieniu układu i następnie zapisywane do pamięci RAM mikrokontrolera. Sensor rozpoczyna pomiar, po przesłaniu ustalonej wartości do rejestru o adresie `0xF4`. Dla temperatury jest to `0x2E`, natomiast dla ciśnienia zależy od wymaganej precyzji wyniku. W systemie wykorzystywany jest tryb standardowy o identyfikatorze `0x74`, z pomiarem trwającym 7,5 ms i uśredniającym dwie próbki. Odczytana temperatura przechowywana jest w rejestrze 16-bitowym, w postaci przemnożonej przez 10. Natomiast ciśnienie, podawane jako wartość całkowita w paskalach (Pa), musi zostać podzielona przez 100, w celu uzyskania jednostki hPa, częściej stosowanej do opisywania stanu pogody. Działania są wykonywane dopiero podczas konwersji do postaci tekstowej, wykorzystując procedurę `printx()`.

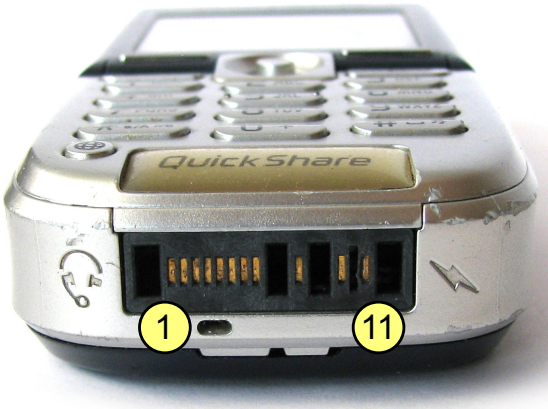
3.4. Komunikacja z telefonem

Najpopularniejszym kablem wykorzystywanym do wymiany danych pomiędzy komputerem a telefonem Sony Ericsson jest DCU-11 pokazany na ilustracji 3.2. We wtyczce znajduje się konwerter UART↔USB, pozwalający na przesył danych bez potrzeby posiadania portu szeregowego w komputerze. Producent często dostarcza wraz z kablem sterowniki konwertera, pozwalające na prawidłową komunikację. Po podłączeniu, możliwa jest transmisja danych z modemem wbudowanym w telefon na przykład przez terminal, wykorzystując wirtualny port szeregowy.

Numeracja pinów złącza telefonu zaczyna się od lewej strony, patrząc od przodu (od strony z wyświetlaczem i klawiaturą), jak zaprezentowano to na ilustracji 3.3. Działanie poszczególnych wyjść zostało objaśnione w tabeli 3.3[2].

[2] Pinouts.ru: *Sony Ericsson R520, R310, R320, T28, T39, T68, T68i, T610, T630 and others cell phones cable connector pinout*, `http://pinouts.ru/CellularPhones-P-W/erics_t28_pinout.shtml` (dostęp: 23.02.2016)

Ilustr. 3.2: Oryginalny kabel DCU-11 z wtyczką USB

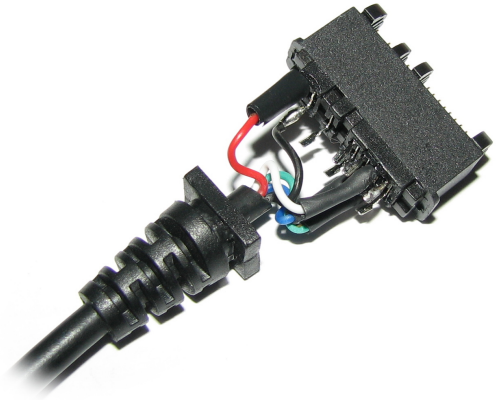


Ilustr. 3.3: Kolejność numeracji pinów złącza Sony Ericsson

W celu dostosowania kabla do komunikacji z systemem wbudowanym, usunięto konwerter USB oraz dokonano niezbędnych zmian we wtyczce od strony telefonu. Początkowy rozkład połączeń – Rx (wysyłanie), Tx (odbieranie), RTS (*Ready To Send*), CTS (*Clear To Send*) i GND (masa) – nie pozwalał na podłączenie zewnętrznego źródła zasilania, a tym samym ładowania aparatu. Ze względu na ograniczoną ilość przewodów zrezygnowano z opcjonalnej sprzętowej kontroli przepływu (linie RTS, CTS) i wykorzystano je do podłączenia napięcia z wyjścia 5V na płytce NUCLEO. Takie podejście dodatkowo udostępnia wyjścia na zewnętrzne źródło audio, które są współdzielone z kontrolą przepływu. Wnętrze wtyczki po zmianach umieszczono na ilustracji 3.4. Kabel czerwony został podłączony do wyjścia nr 11 (Vcc) z początkowego pinu 2 (RTS), natomiast zielony pozostał wolny, do przyszłego zastosowania.

Pin	Oznaczenie	Opis
1	ATMS	Audio do telefonu
2	AFMS/RTS	Audio z telefonu lub Ready To Send
3	CTS	Clear To Send
4	DTMS (Rx)	Dane wysyłane do telefonu (*to mobile*)
5	DFMS (Tx)	Dane odbierane z telefonu (*from mobile*)
6	ACC in	Sterowanie akcesoriami zewnętrznymi
7	ACC out	Sterowanie akcesoriami zewnętrznymi
8	AGND	Masa audio
9	Flash	Pin serwisowy
10	DGND	Masa cyfrowa
11	Vcc	Ładowanie telefonu +5 V

Tab. 3.3: Rozkład złącza telefonu Sony Ericsson



Ilustr. 3.4: Wtyczka kabla DCU-11 po modyfikacjach

3.4.1. Protokół

Komunikacja z telefonem odbywa się asynchronicznie, pełnym dupleksem, przy wykorzystaniu linii sygnałowych modułu `USART6`: Tx do `PA11` i Rx do `PA12`. Napięcie podciągane jest do poziomu 3,3 V. Stan wysoki reprezentuje binarne `0`, a niski 1. Prędkość transmisji jest automatycznie wykrywana przez telefon w momencie rozpoczęcia nadawania. Przeważnie może się ona mieścić w zakresie od 300 do 115200 bps. W tworzonym systemie, parametry połączenia ustalono na: 9600 bps, 8 bitów, brak parzystości, 1 bit stopu. Do komunikacji wykorzystywanie są komendy AT, a cała lista wraz ze szczegółami i obsługiwanymi parametrami przez telefony Sony Ericsson, znajduje się w specyfikacji dla deweloperów [11].

Polecenia zaczynają się od znaków „AT”, a zatwierdzane są znakiem powrotu karetki (*ang. CR – Carriage Return*) o wartości ASCII/IRA 13. Wyjątkiem jest „A/” służące do powtórzenia ostatniej komendy i niewymagające znaku powrotu karetki. Wśród poleceń znajdują się podstawowe, opisane przez standard **ITU** oraz zdefiniowane przez producentów modemów, jak na przykład część rozpoczynających się od „AT+C”, będących rozszerzeniami dla telefonii komórkowej [9]. Jeżeli komenda przyjmuje parametry, są one przeważnie podawane po znaku równości (=) i oddzielane przecinkiem (,). Ciągi znaków muszą być zawarte między cudzysłowami (”). Można również pomijać parametry opcjonalne poprzez niepodawanie żadnego argumentu – wtedy zostanie użyta wartość domyślna.

W przypadku błędu, spowodowanego na przykład nieobsługiwanym poleceniem lub nieprawidłowymi parametrami, modem telefonu odsyła kod „ERROR” wraz z poprzedzającymi znakami nowej linii CR i LF (*ang. Line Feed*) o wartości 10. Pozwala to na uzyskanie czytelnego wyjścia w przypadku komunikacji poprzez terminal tekstowy. Jeżeli żądane polecenie zostanie przyjęte prawidłowo, telefon zwraca odpowiedź zakończoną ciągiem „OK”.

Podłączony aparat może również przysyłać asynchroniczne kody odpowiedzi **URC** (*ang. Unsolicited Result Code*), tj. dane niepowiązane z wysłanym żądaniem, tylko spowodowane zewnętrznym zdarzeniem, na przykład przychodzącym połączeniem lub odebraniem nowej wiadomości SMS. Opracowywany system powinien prawidłowo obsługiwać takie sytuacje. Z tego powodu, w komunikacji wykorzystano opcję echa nadawanego przez telefon. Każdy prawidłowy znak wysłany do modemu zostaje natychmiast odesłany. To pozwala dokładnie zsynchronizować wyjście i zadecydować czy otrzymana odpowiedź jest związana bezpośrednio z wysłanym poleceniem.

Obsługa komunikacji realizowana jest w osobnym zadaniu PhoneTask i procedurze obsługi przerwania (*ang. ISR – Interrupt Service Routine*) podzespołu USART6 generowanego w momencie odebrania bajtu (flaga RXNE) oraz po wysłaniu (flaga TXE). Domyślnie otrzymane odpowiedzi traktowane są jako URC, a ich obsługa następuje po otrzymaniu ciągu CRLF. Podobnie dzieje się z danymi odbieranymi przed uzyskaniem echa transmitowanego polecenia. Jeżeli echo zostanie odebrane prawi-

dłowo, dalsza część odpowiedzi jest przetwarzana, aż do uzyskania ciągu „OK” lub „ERROR”. Jeżeli system nie otrzyma ich w ciągu kilku sekund, traktowane jest to jako błąd limitu czasu (*ang. timeout*). Telefon buforuje kody URC, w trakcie przesyłania poleceń i wyśle je po zakończeniu przetwarzania związanego z aktualnie transmitowaną komendą. Z tego powodu, po każdym wysłanym poleceniu, odczekiwane jest około 150 ms, przed wysłaniem kolejnego, w celu odebrania ewentualnych odpowiedzi.

3.4.2. Wiadomości tekstowe SMS

Standard AT przewiduje dwa tryby przesyłania wiadomości: tekstowy i **PDU** (*ang. Protocol Data Unit*). Pierwszy z nich jest wygodniejszy dla człowieka, ponieważ zawiera wszystkie dane w czytelnym formacie, jednak obsługuje go mniejsza część modemów i telefonów [12]. Drugi tryb jest bezpośrednim przełożeniem niskopoziomowego protokołu wiadomości. Jednocześnie jego możliwości są znacznie bardziej uniwersalne – pozwala m.in. na przesył danych binarnych, a nawet stosowanie formatów nieobsługiwanych przez modem. Wykorzystane telefony w pracy inżynierskiej obsługują tylko tryb **PDU**. Jest to standard binarny, a ponieważ przebieg komunikacji AT powinien być czytelny i umożliwiać łatwy sposób wprowadzania terminalem przez człowieka, dane przesyłane są jako ciągi znaków ASCII, zakodowanych bajtów w systemie szesnastkowym. Ramka danych typowej odebranej wiadomości SMS prezentuje się następująco [13]:

1) *SMSCA* – adres (numer) Centrum Usług SMS:
   1.1) długość całego pola z adresem w oktetach *(1 bajt)*;
   1.2) rodzaj adresu *(1 bajt)*;
   1.3) wartość adresu w kodowaniu BCD *(rozmiar zmienny)*.
2) *First Octet* – rodzaj oraz zawartość wiadomości *(1 bajt)* – wartość `0x04` określa funkcję SMS-DELIVER, tj. wiadomość otrzymaną z Centrum Usług;
3) *Originating Address (OA)* – numer nadawcy:
   3.1) długość adresu w cyfrach *(1 bajt)*;
   3.2) rodzaj adresu *(1 bajt)*;

3.3) wartość adresu nadawcy *(do 12 bajtów)*.

4) *Protocol Identifier (PID)* – identyfikator protokołu *(1 bajt)* – wartość `0x00` określa zwykłą wiadomość SMS;
5) *Data Coding Scheme (DCS)* – schemat kodowania danych *(1 bajt)* – wartość `0x00` dla 7-bitowego kodowania GSM;
6) *Time-Stamp (SCTS)* – znacznik czasu wiadomości *(7 bajtów)* – kodowanie BCD w formacie (pola po 2 cyfry): rok, miesiąc, dzień, godzina, minuta, sekunda, strefa czasowa podana w kwadransach różnicy do GMT;
7) *User Data (UD)* – długość danych *(1 bajt)* + dane *(do 140 bajtów)* – właściwa treść wiadomości zakodowana wg wybranego schematu DCS.

Numery telefonów są formatowane zgodnie ze standardem ITU-T E.164[3]. Najczęściej wykorzystywanymi typami adresów są `0x91` oznaczający numer międzynarodowy zawierający na początku kod kraju (dla Polski 48 – początkowy plus jest pomijany) oraz `0x81` będący numerem lokalnym, działającym tylko w obrębie aktualnego kraju. Adresy przekazywane w ramce PDU mają postać zakodowaną w BCD z półbajtami (*ang. nibble*) zamienionymi miejscami. Ponadto w przypadku nieparzystej liczby cyfr, numer dopełniany jest wartością `0xF`. To oznacza, że na przykład adres +48123 zostanie zapisany w postaci „`8421F3`”. W podobny sposób kodowane jest pole znacznika czasu. Wartość strefy dla czasu środkowoeuropejskiego (CET = GMT+01:00), wykorzystywanego w Polsce (czas zimowy), wynosi `0x04` i jest reprezentowane przez ciąg w postaci „`40`”. Treść wiadomości standardowo kodowana jest w septetach (7-bitowych pakietach) według alfabetu GSM, który częściowo pokrywa się ze standardem ASCII/IRA, w szczególności litery alfabetu łacińskiego oraz cyfry [14]. Pozwala to na przesyłanie prostych wiadomości bez konieczności implementacji konwersji kodowania. W pojedynczej wiadomości można zmieścić do 160 znaków w 140 bajtach.

Ramka wiadomości wysyłanych częściowo różni się od odebranych. Pole adresu Centrum Usług może zostać pominięte, podając wartość `0x00` – telefon korzysta

[3] ITU-T: *E.164: The international public telecommunication numbering plan*, 11.2010, `https://www.itu.int/rec/T-REC-E.164/en` (dostęp: 11.03.2016)

wtedy z numeru przypisanego przez operatora aktualnie podłączonej sieci:

1) *SMSCA* – adres Centrum Usług SMS;
2) *First Octet* – rodzaj wiadomości i format terminu ważności *(1 bajt)* – wartość `0x11` określa funkcję SMS-SUBMIT, tj. wiadomość wysyłaną do Centrum Usług oraz format ważności relatywny;
3) *Message Reference (MR)* – identyfikator wysyłanej wiadomości, pozwalający powiązać ją z raportami Centrum Usług *(1 bajt)* – nie każdy telefon obsługuje identyfikatory użytkownika i należy wtedy podać wartość „`0x00`”;
4) *Destination Address (DA)* – numer odbiorcy – format taki sam jak w *Originating Address*;
5) *Protocol Identifier (PID)*;
6) *Data Coding Scheme (DCS)*;
7) *Validity Period (VP)* – okres ważności wiadomości *(relatywny: 1 bajt)* [13] – przykładowo wartość `143` dla 12 godzin od wysłania, `167` dla 24;
8) *User Data (UD)*.

Aby system mógł natychmiast reagować na przysyłane SMSy, aktywowane są kody URC związane z nowymi wiadomościami [11, 15]. Odbywa się to za pomocą komendy „`AT+CNMI=,1`” (*New Message Indication*). Pierwszy parametr określający metodę buforowania odpowiedzi jest pomijany, ponieważ Sony Ericsson obsługuje tylko jeden domyślny sposób. Wartość `1` powoduje, że zostanie wyświetlone miejsce zapisu przychodzącej wiadomości, zawierające rodzaj pamięci (telefon lub karta SIM) oraz numer pozycji. Przykładowo komunikat „`+CMTI: "ME",3`” informuje o nowej wiadomości, zapisanej w pamięci telefonu (*ang. Mobile Equipment*) pod indeksem 3. W celu pobrania jej treści, należy upewnić się, że została wybrana odpowiednia pamięć SMS poleceniem „`AT+CPMS="ME","ME","ME"`” (*Preferred Message Storage*). Następnie wysyłane jest „`AT+CMGR=3`” (*Read Message*) z otrzymanym wcześniej indeksem. Telefon zwraca komunikat „`+CMGR: 0,`”, wraz z długością całej wiadomości w oktetach, po której znajduje się właściwa zawartość ramki PDU w szesnastkowej reprezentacji ASCII. Jeżeli proces komunikacji przebiegł pomyślnie, SMS może zostać usunięty z telefonu poleceniem „`AT+CMGD=3`”, aby zapobiec przepełnieniu pamięci.

Wysyłanie wiadomości odbywa się komendą „AT+CMGS=42”, gdzie 42 to przykładowa ilość oktetów PDU, które są podawane po znaku powrotu karetki CR, w takim samym formacie jak przy odbieraniu. Zatwierdzenie podanej ramki dokonywane jest klawiszem [Ctrl+Z], tj. znakiem o wartości ASCII 26. W trakcie wysyłania wiadomości modem przestaje odpowiadać, a trwa to kilkanaście sekund, dlatego w przypadku tego polecenia limit czasowy został zwiększony do 30 s, aby zapobiec niepotrzebnym błędom. Jeśli SMS zostanie przesłany prawidłowo, telefon odpowiada „OK”, a w przeciwnym wypadku „ERROR”.

3.5. Czasowe wysyłanie powiadomień

W celu umożliwienia wysyłania pomiarów stacji co określony przez użytkownika czas, wykorzystano wbudowany w mikrokontroler zegar czasu rzeczywistego **RTC**. Aktualny czas jest pobierany z podłączonego telefonu za pomocą polecenia „AT+CCLK?”. Odpowiedź w postaci „+CCLK: ”YY/MM/DD,hh:mm:ss±zz”” jest parsowana i przesyłana do rejestrów zegara. Wartość „±zz” oznacza strefę czasową podaną w różnicy kwadransów od czasu GMT, podobnie jak w polu 6) *Time-Stamp (SCTS)* datagramu PDU opisanego w rozdziale 3.4.2. Aby wskazywana godzina była prawidłowa, telefon został ustawiony w taki sposób, żeby pobierał czas od operatora sieci GSM – w przypadku modelu K300i funkcja nazywa się „Auto time zone”. Sam układ RTC mikrokontrolera jest taktowany zewnętrznym rezonatorem kwarcowym X2 znajdującym się na płytce NUCLEO [3] o częstotliwości 32,768 kHz.

Godzina automatycznego wysyłania ustawiana jest w rejestrze pierwszego alarmu (ALRA) podzespołu RTC według ustawionego przez użytkownika opóźnienia. Przerwanie wywołane w momencie zrównania wskazania zegara powoduje wysłanie wiadomości SMS ze wskazaniami stacji i przestawienie godziny alarmu, w celu uzyskania regularnego powiadamiania co ustalony czas.

3.6. Terminal tekstowy i ustawienia

Jak zostało wspomniane w rozdziale 2.1, płytka startowa NUCLEO umożliwia wykorzystanie programatora jako konwerter UART↔USB. Umożliwia to podłączenie systemu do komputera posiadającego wejście USB i komunikację za pomocą wirtualnego portu szeregowego. Konwerter jest wewnętrznie przyłączony do wyjść modułu `USART2`: `PA2` i `PA3`. Za komunikację odpowiada zadanie `TerminalTask`, które pobiera dane z bufora i wykonuje działania zlecone przez użytkownika. Konsola umożliwia konfigurację systemu m.in. w zakresie trybu działania, powiązanego numeru telefonu, czy kodu kraju. Zapis ustawień odbywa się w pamięci trwałej mikrokontrolera, za pomocą mechanizmu programowania w aplikacji (*ang. IAP – In Application Programming*). Wykorzystana pamięć Flash jest podzielona na sektory o różnej wielkości: 4× 16 kB, 1× 64 kB i 3× 128 kB. Czyszczenie danych może odbywać się tylko w całym sektorze, dlatego z powodu niewielkiej ilości ustawień wybrano sektor 16 kB. Aby zapewnić wolną przestrzeń, zmodyfikowano skrypt konsolidatora (plik `LinkerScript.ld`), umieszczając w pierwszym sektorze tylko dane bootloadera (m.in. konfiguracja oscylatora, adresy procedur przerwań). Cały program przeniesiono do regionu 64 kB, zostawiając najmniejsze sektory wolne, na zapis ustawień.

3.7. Zarządzanie energią

Modyfikacja kabla połączeniowego opisana w rozdziale 3.4 umożliwiła zasilanie podłączonego telefonu, ładując znajdującą się w nim baterię. Dzięki temu możliwe jest uruchomienie wyłączonego aparatu – napięcie zasilania powoduje przejście telefonu w tryb ładowania, co następnie umożliwia przesyłanie komend AT. Zaraz po włączeniu całego systemu i inicjalizacji poszczególnych podzespołów, następuje sprawdzenie stanu telefonu poleceniem „`AT+CFUN?`”. Otrzymana odpowiedź „`+CFUN: 1`” oznacza całkowite włączenie aparatu i gotowość na komunikację z siecią komórkową. W przeciwnym wypadku wysyłane jest „`AT+CFUN=1`” przełączające aparat w żądany tryb działania.

Stan baterii jest kontrolowany cyklicznie poleceniem „AT+CBC?”. W odpowiedzi znajdują się dwa parametry. Pierwszy określa stan zasilania: „0” wskazuje na brak zewnętrznego zasilania, a „1” na ładowanie baterii. Zastosowanie takiego rozwiązania pozwala również na działanie całego systemu korzystając z baterii telefonu i wykrycie awarii głównego źródła zasilania.

W celu oszczędzania energii, mikrokontroler posiada 3 tryby uśpienia, podczas których nie są wykonywane instrukcje. Pierwszy z nich – „Sleep mode” – wyłącza główne jądro procesora, pozostawiając działanie podzespołów i oczekując na wywołanie przerwania przez któreś z nich. Wykonywane jest to poleceniem „WFI” (*Wait For Interrupt*), natomiast wybudzenie z uśpienia zajmuje od 4 do 6 cykli procesora, po których następuje przejście do procedury obsługi odpowiedniego przerwania. Drugi tryb „Stop mode”, dodatkowo wyłącza większość linii zegarowych, pętle PLL i oscylatory, co powoduje zatrzymanie większości podzespołów, oprócz m.in. RTC, pamięci RAM oraz wejść mikrokontrolera. Dostępne jest również opcjonalne odłączenie zasilania pamięci Flash mikrokontrolera, jednak wiąże się to z wydłużonym czasem budzenia. Usypianie realizuje się podobnie jak w „Sleep mode”, z dodatkowym ustawieniem bitu SLEEPDEEP rejestru *System Control*. Najbardziej oszczędnym trybem jest „Standby mode”, które powoduje wyłączenie zasilania pamięci RAM (pomijając rejestry podtrzymywane w *Backup Domain*), a przez to utratę jej zawartości. Wejście w uśpienie uzyskuje się przez ustawienie bitu PDDS. Wybudzenie może spowodować reset MCU, zmiana stanu pinu WKUP (Low-power wakeup) lub alarm RTC.

W stworzonym systemie zastosowano pierwszy tryb „Sleep mode”, który jest aktywowany, gdy wszystkie zadania systemu operacyjnego są uśpione lub zablokowane. Nie zdecydowano się na wykorzystanie bardziej oszczędnych trybów, ze względu na brak możliwości zasilania modułu USART. To uniemożliwiałoby natychmiastową reakcję na odpowiedzi przesyłane przez telefon, takie jak otrzymanie wiadomości lub połączenie przychodzące.

Rozdział 4

Podręcznik użytkownika

4.1. Uruchomienie

Układ zasilany jest napięciem 5 V pochodzącym ze źródła podłączonego pod gniazdo mini-USB. Minimalna wydajność prądowa powinna wynosić 500 mA, aby umożliwić bezproblemowe zasilanie podłączonego telefonu. Przykładem takiego źródła jest komputer, ładowarka USB lub przenośna bateria (Power Bank). Po uruchomieniu następuje inicjalizacja modułów znajdujących się na płytce oraz wykonane jest badanie ich sprawności.

4.2. Konfiguracja i diagnostyka

W przypadku podłączenia układu do komputera lub innego urządzenia obsługującego sterownik programatora, możliwa jest komunikacja poprzez terminal tekstowy. Wymagane ustawienia połączenia: **115200/8-N-1** (115200 bps, 8 bitów danych, brak parzystości, 1 bit stopu). Wyświetlenie ekranu z informacją o systemie oraz dostępnymi opcjami konfiguracji, dokonywane jest klawiszem [ESC]. Jeżeli wyjście diagnostyczne (*ang. debug output*) jest włączone (klawisz [D]), użytkownik będzie informowany na bieżąco o zmianach stanu poszczególnych modułów systemu oraz przebiegu komunikacji z podłączonym telefonem. Aktualny stan systemu, w tym wskazania czujników wyświetlane są po naciśnięciu przycisku [S]. Przykładowe wyjście zostało przedstawione w listingu 4.1.

```
******************************
 Current status: [OK]
   > Storage: [OK]
   > Phone: [OK]
     > Battery: 100 % (charging)
   > Clock: [OK]
     > 02.03.2016 21:37:13
     > Auto SMS at: 21:45:00
   > Sensors:
     > DHT-22: [OK]
       > T: 22.6 *C
       > RH: 46.6 %
     > BMP180: [OK]
       > T: 22.3 *C
       > P: 983.06 hPa
******************************
```

Listing 4.1: Przykładowy raport stanu systemu

Wyświetlane są takie informacje jak poziom naładowania baterii, godzina pobrana z telefonu, czas kolejnego powiadomienia SMS oraz pomiary czujników. Zaznaczone są również awarie poszczególnych podzespołów systemu jako „`[Error]`” lub ich prawidłowe działanie „`[OK]`”.

Konsola pozwala na dostosowanie powiązanego numeru telefonu (klawisz [P]), na który mają być przesyłane raporty stanu. W przypadku zmiany kraju, należy także zadbać o prawidłowy prefiks narodowy (*ang. country code*) (klawisz [C]), który dla Polski wynosi +48. Po dostosowaniu ustawień, można dokonać ich zapisu w pamięci trwałej klawiszem [!] ([Shift+1]). W przeciwnym razie, zmiany zostaną utracone po utracie zasilania lub restarcie systemu.

4.3. Wiadomości SMS z aktualnym stanem

Po skonfigurowaniu numeru telefonu służącego do komunikacji, wysłanie z niego wiadomości tekstowej o treści „STATUS”, spowoduje odesłanie SMSa zawierającego aktualne wskazania czujników pogodowych. Wielkość znaków w treści nie ma wpływu na działanie. Wiadomości kodowane inaczej niż 7-bitowym alfabetem GSM będą odrzucane – nie mogą zawierać znaków narodowych i symboli Unicode. Istnieje również możliwość ustawienia automatycznego wysyłania wiadomości co określony czas. W celu skonfigurowania takiej możliwości, należy połączyć się

z terminalem tekstowym w sposób opisany w rozdziale 4.2 i wybrać przycisk [A]. Następnie system zapyta o interwał czasowy w minutach, co ile ma być wysyłany raport. Podanie wartości „`0`” wyłącza automatyczne przesyłanie.

Treść wiadomości zawiera aktualne wskazania czujników oraz ewentualne nieprawidłowości w działaniu podzespołów systemu. Została ona umieszczona w listingu 4.2.

```
02.03.2016 21:33:02
Temp: 23.7 *C
RH: 43.7 %
Pres: 982.91 hPa
```

Listing 4.2: Przykładowy SMS z pomiarami

Należy zauważyć, że podawane ciśnienie p zależy także od wysokości położenia stacji. W celu wyliczenia wartości p_0 na poziomie morza można wykorzystać wzór kompensujący (4.1) z dokumentacji czujnika [8]. Wysokość h podawana jest w metrach.

$$p_0 = \frac{p}{\left(1 - \frac{h}{44330}\right)^{5{,}255}} \tag{4.1}$$

Rozdział 5

Wnioski i ocena działania

Stworzony system w ramach tej pracy inżynierskiej jest przykładem stacji pogodowej z możliwością komunikacji zdalnej poprzez sieć telefonii komórkowej. Do pomiarów wykorzystane zostały dwa czujniki, które pozwalają na odczyt temperatury, wilgotności i ciśnienia powietrza z dokładnością wystarczającą do zastosowań amatorskich. Przesył danych zrealizowano za pomocą komend AT Hayesa transmitowanych do podłączonego telefonu Sony Ericsson. Przetestowanie działania na dwóch różnych modelach aparatów oraz utrzymanie zgodności protokołu ze specyfikacją wydaną przez producenta, pozwalają na stwierdzenie, że stacja powinna działać również z innymi urządzeniami tej marki. Stałe podłączenie telefonu do zewnętrznego źródła zasilania, pozwala na ładowanie znajdującej się wewnątrz baterii i zapobiega jej całkowitemu rozładowaniu. Dodatkowo w przypadku wyłączenia się aparatu, pozwala to na ponowne uruchomienie za pomocą przesyłanej komendy, co byłoby niemożliwe bez zewnętrznego zasilania. Te argumenty potwierdzają wstępne założenie, że telefon komórkowy starszej generacji może zostać wykorzystany w roli modemu, zachowując jego funkcjonalność i umożliwiając obniżenie kosztów całego systemu.

Powstała stacja może zostać wykorzystana na przykład do monitorowania parametrów meteorologicznych plantacji roślinnej w szklarni, czy też na otwartej przestrzeni w ogródku działkowym. Pozwala to na wcześniejsze wykrycie niekorzystnych warunków, bez potrzeby przemieszczania się w inne miejsce. Zdarzeniem wymagającym dokonania działań zapobiegających mogłaby być niedostateczna wil-

gotność powietrza lub nadmierna temperatura, a przez to konieczność nawilżenia gleby.

System został zaprojektowany w taki sposób, że rozszerzenie jego funkcjonalności, poprzez dodanie kolejnych sensorów lub elementów realizujących określone działanie (np. przekaźnika sterującego nawilżaczem lub nagrzewnicą) nie powinno stanowić problemu. Pod względem sprzętowym, uniwersalna płytka rozszerzająca opisana w rozdziale 3.1 posiada wolne miejsce na dołączenie podzespołów, a wyjścia goldpin zestawu **NUCLEO**, zapewniają łatwy sposób podłączenia wymaganego zasilania oraz wyjść mikrokontrolera. Natomiast wykorzystanie systemu operacyjnego umożliwia bezproblemowe dodanie oprogramowania przez implementację kolejnego zadania-wątku, czy procedury obsługi przerwania. Stworzona stacja pogodowa mogłaby stać się bardziej uniwersalna, implementując obsługę innych telefonów komórkowych, jak chociażby wspomniane w rozdziale 2.6 protokoły **MBUS** i **FBUS**.

Z drugiej strony, taki system komunikujący się przez łączność telefoniczną może zostać wykorzystany także do innych celów. Przykładowym zastosowaniem jest informowanie o wykryciu niepożądanego ruchu przez fotokomórkę, z jednoczesną możliwością nasłuchiwania otoczenia przez wbudowany w telefon mikrofon.

Bibliografia

[1] STMicroelectronics: *STM32F401xD STM32F401xE Datasheet*, (DocID025644 Rev 3), 01.2015, http://www.st.com/web/en/resource/technical/document/datasheet/DM00102166.pdf (dostęp: 15.02.2016)

[2] STMicroelectronics: *STM32F401xB/C and STM32F401xD/E Reference Manual*, RM0368, (DocID025350 Rev 4), 05.2015, http://www.st.com/web/en/resource/technical/document/reference_manual/DM00096844.pdf (dostęp: 15.02.2016).

[3] STMicroelectronics: *STM32 Nucleo-64 boards User manual*, UM1724, (DocID025833 Rev 9), 08.2015, http://www.st.com/web/en/resource/technical/document/user_manual/DM00105823.pdf (dostęp: 15.02.2016)

[4] STMicroelectronics: *Description of STM32F30xx/31xx Standard Peripheral Library*, UM1581, (DocID023800 Rev 1), 10.2012, http://www.st.com/web/en/resource/technical/document/user_manual/DM00068049.pdf (dostęp: 15.02.2016)

[5] Aosong: *Temperature and humidity module AM2302 Product Manual*, http://akizukidenshi.com/download/ds/aosong/AM2302.pdf (dostęp: 15.02.2016)

[6] Aosong: *Digital-output relative humidity & temperature sensor/module AM2303*, http://kamami.pl/dl/dht22_ds.pdf (dostęp: 15.02.2016)

[7] Maxim Integrated: *1-Wire Communication Through Software*, APP 126, 30.05.2002, https://www.maximintegrated.com/en/app-notes/index.mvp/id/126 (dostęp: 15.02.2016)

[8] Bosch: *BMP180 Digital pressure sensor Data sheet*, Revision 2.5, BST-BMP180-DS000-09, (Reference 0 273 300 244), 05.04.2013, http://cdn.sparkfun.com/datasheets/Sensors/Pressure/BMP180.pdf (dostęp: 15.02.2016)

[9] ITU-T: *V.250 Recommendation: Serial asynchronous automatic dialling and control*, 07.2003, https://www.itu.int/rec/T-REC-V.250/en (dostęp: 15.02.2016)

[10] 3GPP: *TS 27.007: AT command set for User Equipment (UE)*, ETSI TS 127 007, V12.11.0, 01.2016, http://www.etsi.org/deliver/etsi_ts/127000_127099/127007/12.11.00_60/ts_127007v121100p.pdf (dostęp: 15.02.2016)

[11] Sony Ericsson: *AT commands for Sony Ericsson phones*, 17th edition, 06.2010, (Publication no. 1206-6103.17), http://dl-developer.sonymobile.com/documentation/DW-65054-dg_at_2006--10_r17a.pdf (dostęp: 15.02.2016)

[12] J. Bogusz: *Moduły GSM w systemach mikroprocesorowych*, Wydawnictwo BTC, Warszawa 2007.

[13] 3GPP: *TS 23.040: Technical realization of the Short Message Service (SMS)*, ETSI TS 123 040, V12.2.0, 10.2014, http://www.etsi.org/deliver/etsi_ts/123000_123099/123040/12.02.00_60/ts_123040v120200p.pdf (dostęp: 27.02.2016)

[14] 3GPP: *TS 23.038: Alphabets and language-specific information*, ETSI TS 123 038, V12.0.0, 10.2014, http://www.etsi.org/deliver/etsi_ts/123000_123099/123038/12.00.00_60/ts_123038v120000p.pdf (dostęp: 27.02.2016)

[15] 3GPP: *TS 27.005: Use of Data Terminal Equipment - Data Circuit terminating Equipment (DTE - DCE) interface for Short Message Service (SMS) and Cell Broadcast Service (CBS)*, ETSI TS 127 005, V13.0.0, 01.2016, http://www.etsi.org/deliver/etsi_ts/127000_127099/127005/13.00.00_60/ts_127005v130000p.pdf (dostęp: 27.02.2016)

Spis ilustracji