

**Akademia Górniczo-Hutnicza  
im. Stanisława Staszica w Krakowie**

---

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii  
Biomedycznej

KATEDRA AUTOMATYKI I INŻYNIERII BIOMEDYCZNEJ



**PRACA INŻYNIERSKA**

**MICHAŁ MAKA**

**SYSTEM POMIARU WARUNKÓW  
ŚRODOWISKOWYCH I METEOROLOGICZNYCH**

PROMOTOR:

dr inż. Marek Stencel

Kraków 2013

## **OŚWIADCZENIE AUTORA PRACY**

OŚWIADCZAM, ŚWIADOMY ODPOWIEDZIALNOŚCI KARNEJ ZA  
POŚWIADCZENIE NIEPRAWDY, ŻE NINIEJSZĄ PRACĘ DYPLOMOWĄ  
WYKONAŁEM OSOBIŚCIE I SAMODZIELNIE, I NIE KORZYSTAŁEM ZE  
ŹRÓDEŁ INNYCH NIŻ WYMIENIONE W PRACY.

.....

PODPIS

**AGH**  
**University of Science and Technology in Krakow**

---

Faculty of Electrical Engineering, Automatics, Computer Science and  
Biomedical Engineering

DEPARTMENT OF AUTOMATICS AND BIOENGINEERING



**ENGINEERING THESIS**

**MICHAŁ MAKA**

**MEASUREMENT SYSTEM OF ENVIRONMENTAL  
AND WEATHER CONDITIONS**

**SUPERVISOR:**  
**Marek Stencel Sc.D**

**Krakow 2013**



## Spis treści

<b>Wstęp.....</b>	<b>6</b>
<b>1. Badanie warunków środowiskowych i meteorologicznych .....</b>	<b>7</b>
<b>2. Elektroniczny system pomiarów .....</b>	<b>8</b>
<b>3. Mikrokomputer BeagleBone Black .....</b>	<b>9</b>
<b>4. Czujnik ciśnienia atmosferycznego BMP085 .....</b>	<b>11</b>
<b>5. Czujnik wilgotności DHT-22 .....</b>	<b>12</b>
<b>6. Magistrale szeregowo .....</b>	<b>16</b>
6.1. Magistrala I <sup>2</sup> C .....	17
6.2. Magistrala One-Wire .....	17
<b>7. Programowanie mikrokontrolerów ARM.....</b>	<b>18</b>
7.1. Używanie bibliotek Linux'a .....	18
7.2. Środowisko programowania .....	18
7.3. Kompilator .....	20
<b>8. Tworzenie aplikacji zbierającej dane pomiarowe .....</b>	<b>21</b>
<b>9. Przechowywanie oraz wyświetlanie wyników .....</b>	<b>22</b>
9.1. Serwer HTTP .....	22
9.2. Baza danych MySQL.....	22
9.3. Interfejs użytkownika .....	22
<b>10. Rezultaty pomiarów .....</b>	<b>23</b>
<b>11. Podsumowanie .....</b>	<b>24</b>
 <b>Bibliografia .....</b>	 <b>25</b>
 <b>Załączniki .....</b>	 <b>26</b>
<b>A. Kod źródłowy aplikacji.....</b>	<b>26</b>

**Wstep**

# 1. Badanie warunków środowiskowych i meteorologicznych

Zbieranie informacji na temat aktualnych danych pogodowych stało się bardzo ważne w dzisiejszych czasach. Różne firmy z dziedziny automatyki, budownictwa, przemysłu itp. potrzebują tych danych, aby wiedzieć jak zorganizować swoją pracę oraz przewidzieć plan na najbliższą przyszłość. Posiadając aktualne pomiary pogodowe można zmienić plan pracy, jeżeli warunki atmosferyczne na to nie pozwalają.

Przykładowo przeprowadzenie naprawy sieci elektrycznej przy bardzo dużej wilgotności powietrza jest stosunkowo niebezpieczne, ponieważ mogą występować przepięcia, prowadzące do całkowitego jej uszkodzenia. Badanie temperatury otoczenia jest niezmiernie potrzebne przy instalacjach, które wymagają określonego przedziału ciepła, stosuje się to np. w chłodniach. Dzięki pomiarom temperatury można sterować regulatorem ogrzewania lub chłodzenia w celu uzyskania optymalnych i rekomendowanych warunków pracy urządzeń przemysłowych.

Na podstawie aktualnych danych pogodowych wraz z historią pomiarów, dzięki zastosowaniu specjalnych algorytmów, można uzyskać z pewnym prawdopodobieństwem prognozę pogody na najbliższą przyszłość. Wyniki przewidywania pogody nigdy nie są idealne, ale pozwalają na przygotowanie się na możliwe warunki atmosferyczne.

Pomiary warunków środowiskowych są nieodłącznym elementem każdego regulatora automatyki, zastosowanego w klimatyzatorach, lodówkach, grzejnikach, nawilżaczach lub osuszaczach powietrza. Na ich podstawie wyliczane jest sterowanie, które następnie jest podawane na element wykonawczy, np. część chłodzącą. Pomiar aktualnej wartości z wartością zadaną jest porównywany nieustannie w cyklu działania urządzenia.

Projekt inteligentnego domu, w którym niemal wszystko jest regulowane automatycznie, wymaga informacji o aktualnych warunkach i na tej podstawie ustawia poziom ogrzewania, nawilżania powietrza.

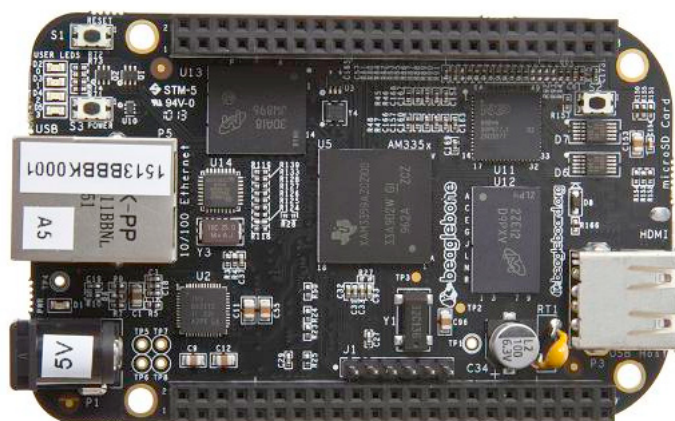
## **2. Elektroniczny system pomiarów**



### 3. Mikrokomputer BeagleBone Black

Projekt inżynierski został zrealizowany, w głównej części, na mikrokomputerze BeagleBone Black. Został on stworzony specjalnie z myślą o programistach OpenSource oraz tych, dla których liczy się niskie zużycie energii. Jest to oparta na procesorze AM335x ARM Cortex-A8, taktowany częstotliwością 1 GHz, płytka developerska, która została wyposażona w 512 MB pamięci RAM, 2 GB pamięci FLASH, akcelerator grafiki 3D. Posiada szereg różnych interfejsów, takich jak: HDMI, USB, Ethernet, czytnik kart microSD. BeagleBone można zasilać na dwa sposoby, pierwszy - poprzez kabel USB podłączony do USB (5V) albo przy użyciu zewnętrznego zasilacza, również 5V. Dla użytkownika zostały również wyprowadzone 96 pinów typu wejście/wyjście.

Na mikrokomputerze można zainstalować i ze swobodą korzystać z najpopularniejszych dystrybucji Linuxa, np. Ubuntu, Debian, Fedora, Arch. Istnieje również możliwość uruchomienia na BeagleBone systemu Android.

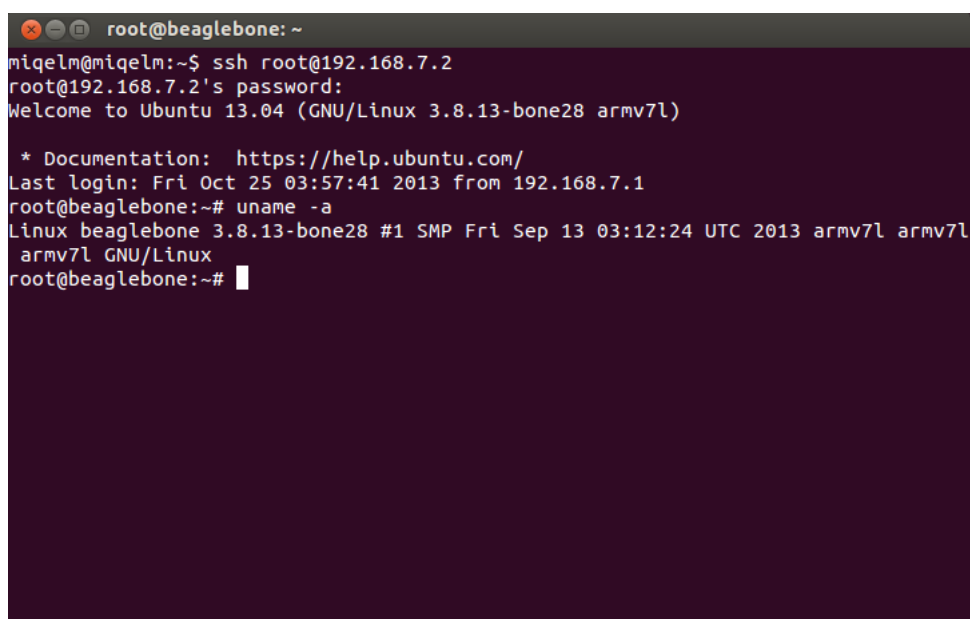


Rysunek 3.1: BeagleBone Black z góry

Po zakupie, domyślnie BeagleBone posiada zainstalowaną dystrybucję Linuxa - Ångström. Z uwagi na większą znajomość innego systemu operacyjnego - Ubuntu, na mikrokontrolerze została zainstalowana właśnie ta dystrybucja w wersji 13.04, dedykowana na platformę ARM Hard Float. Platforma ta posiada zaimplementowaną sprzętową obsługę liczb zmiennoprze-

cinkowych. Obraz systemu operacyjnego oraz instrukcję jego zainstalowania za pomocą karty pamięci microSD można znaleźć na głównej stronie projektu ARM hf: [www.armhf.com](http://www.armhf.com).

BeagleBone został podłączony do komputera z zainstalowanym środowiskiem programistycznym przy użyciu portu USB. Po zainstalowaniu odpowiednich sterowników, które znajdują się na oficjalnej stronie producenta tej płytki oraz pamięci FLASH, port ten jest wykrywany jako interfejs sieciowy i tworzona jest sieć łącząca komputer z mikrokontrolerem. Domyślne ustawienia sprawiają, że do BeagleBone'a można podłączyć się przy użyciu protokołu SSH, łącząc się z adresem: 192.168.7.2. Po poprawnym zalogowaniu się do płytki poprzez program ssh, dostępny na Ubuntu, otrzymamy ekran podobny do tego poniżej:

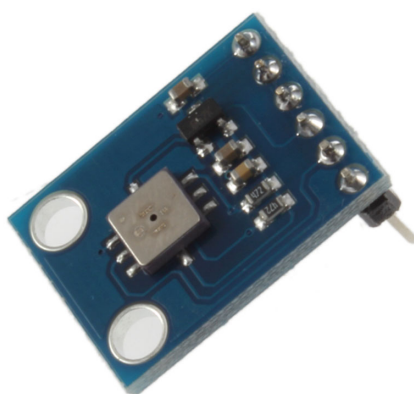


```
root@beaglebone: ~
miqelm@miqelm:~$ ssh root@192.168.7.2
root@192.168.7.2's password:
Welcome to Ubuntu 13.04 (GNU/Linux 3.8.13-bone28 armv7l)

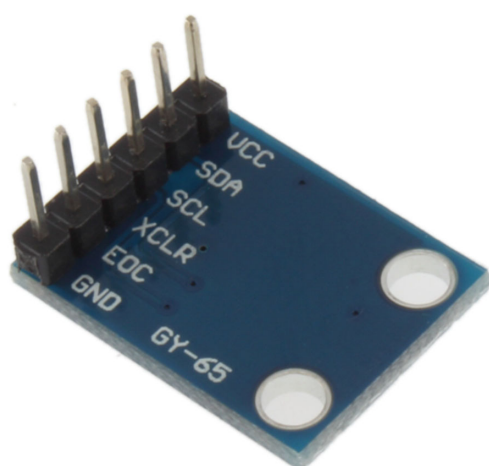
 * Documentation:  https://help.ubuntu.com/
Last login: Fri Oct 25 03:57:41 2013 from 192.168.7.1
root@beaglebone:~# uname -a
Linux beaglebone 3.8.13-bone28 #1 SMP Fri Sep 13 03:12:24 UTC 2013 armv7l armv7l
armv7l GNU/Linux
root@beaglebone:~#
```

Rysunek 3.2: Zrzut ekranu z konsoli

## 4. Czujnik ciśnienia atmosferycznego BMP085



Rysunek 4.1: Czujnik ciśnienia BMP085 - widok z dołu



Rysunek 4.2: Czujnik ciśnienia BMP085 - widok z góry

## 5. Czujnik wilgotności DHT-22

### Opis

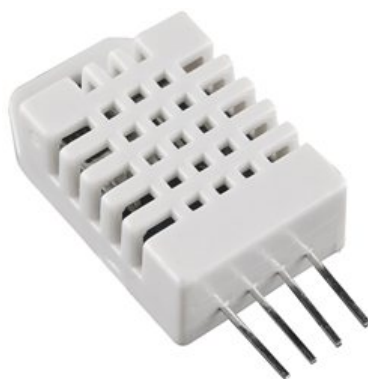
DHT-22 jest cyfrowym oraz fabrycznie skalibrowanym czujnikiem względnej wilgotności i temperatury. Został stworzony w oparciu o 8 bitowy mikrokontroler, do którego podłączone są sensory. Małe rozmiary, stabilność pomiarowa, możliwość zainstalowania czujnika na duże odległości oraz niewielki pobór energii czyni go bardzo dobrym rozwiązaniem, nawet w ciężkim warunkach.

Czujnik posiada swój własny sposób uzyskiwania pomiarów. Jest on podobny do opisanej wcześniej magistrali 1wire. Do prawidłowego działania urządzenia potrzebne jest zasilanie, linia danych, poprzez którą następuje komunikacja oraz uziemienie. Należy również podłączyć do linii danych oraz zasilania rezystor podciągający o oporze około 3,3 k $\Omega$ .

### Dane charakterystyczne

Jak podaje specyfikacja czujnika, może być on zasilany napięciem od 3,3 do 6 V, posiada zdolności pomiarowe: wilgotność w zakresie 0 do 100 % (z dokładnością  $\pm 2\%$ ) oraz temperatury od -40 do 80 stopni Celsjusza ( $\pm 0,5$  stopnia).

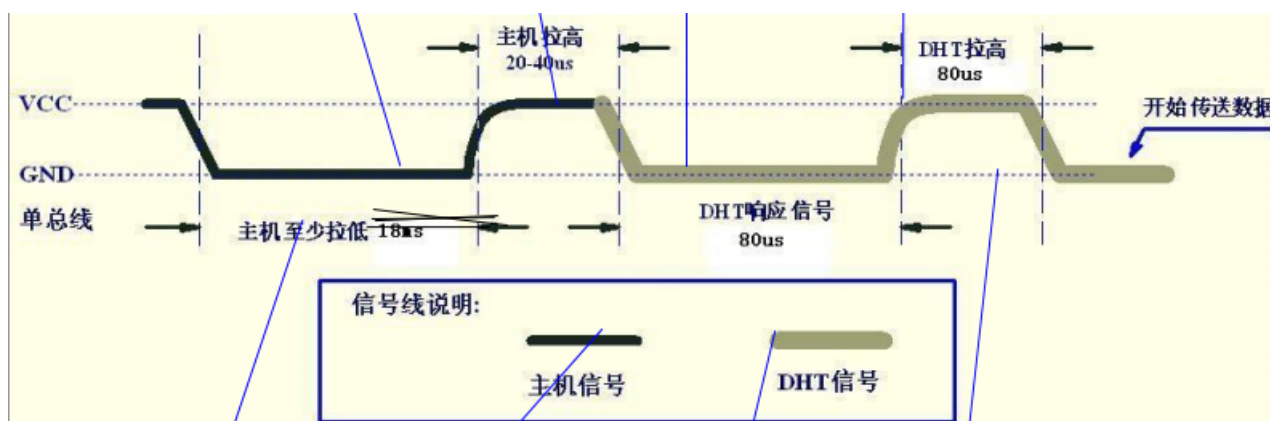
Wygląd czujnika został przedstawiony na poniższym zdjęciu:



Rysunek 5.1: Czujnik wilgotności DHT-22

## Pobieranie danych z czujnika

Zgodnie ze specyfikacją, załączoną do DHT-22, linia danych jest wolna, jeżeli jest na niej ustawiony stan wysoki. Czujnik, gdy nie jest używany, jest ustawiony w tryb spoczynku i niskiego poboru prądu. Poprzez wysłanie sygnału startu jest on wybudzany do trybu pracy i wysyła dane. Aby rozpocząć pomiar trzeba wysłać sygnał startu, należy to uczynić poprzez zwarcie do masy linii danych na co najmniej 18 ms. Po tym czasie czujnik zgłasza swoją obecność w układzie poprzez wystawienie logicznej jedynki na około 20-40  $\mu\text{s}$ , następnie linia danych jest zwalniana oraz ponownie podciągana do zasilania, każdy z tych czynności trwa 80  $\mu\text{s}$ . Poniższy rysunek przedstawia inicjalizację czujnika DHT-22. Po zgłoszeniu swojej obecności w



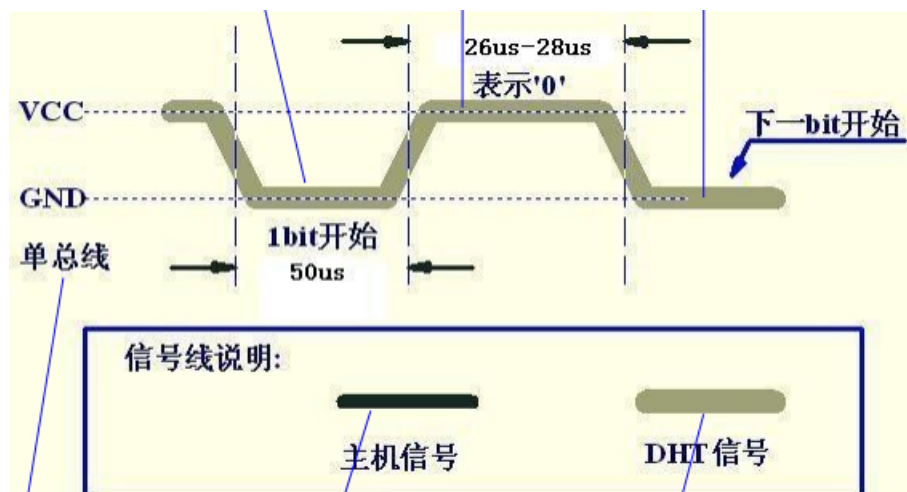
Rysunek 5.2: Inicjalizacja czujnika

układzie, czujnik zaczyna wysyłać 40 bitów danych, pierwsze 16 bitów odpowiada za ciśnienie, następne 16 to temperatura oraz końcowe 8 bitów służy jako suma kontrolna.

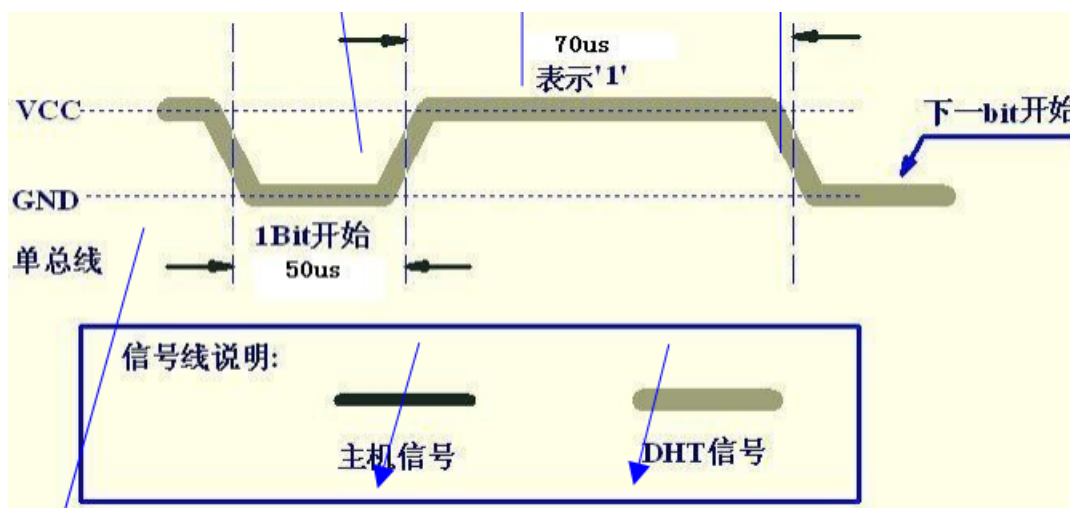
Aby przekonwertować odebrane dane na wskazania czujnika, należy obie liczby 16-bitowe, odpowiadające wilgotności oraz temperaturze, podzielić przez 10. W ten sposób otrzymujemy pomiary z dokładnością do jednego miejsca po przecinku. Po odebraniu danych należy przede wszystkim sprawdzić, czy to co zostało otrzymane jest prawidłowe. Należy pierwsze 4 bajty dodać do siebie oraz sprawdzić, czy 8 młodszych bitów tej wartości są równe z ostatnim odebranym bajtem. Jeżeli tak, procedura odebrania została zakończona sukcesem. W przeciwnym wypadku należy spróbować ponownie pobrać dane z czujnika.

Sposób kodowania informacji na linii danych przez DHT-22 jest następujący: każda wartość logiczna (0 oraz 1) jest reprezentowana przez podpięcie linii do zasilania na ustaloną chwilę czasową. Bit o wartości 0 jest wysyłany jako wysoki poziom przez czas 26-28  $\mu\text{s}$ , natomiast czas trwania dla bitu 1 wynosi 70  $\mu\text{s}$ . Pomiedzy nadaniem odpowiedniego bitu, występuje przerwa - podłączenie linii danych do uziemienia na około 50  $\mu\text{s}$ .

Przebiegi poniżej pokazują wysyłanie danych:



Rysunek 5.3: Wysłanie logicznego "0"



Rysunek 5.4: Wysłanie logicznej "1"

Wysyłanie danych jest zakończone, jeżeli linia danych jest podłączona do zasilania oraz nie zmienia się jej stan. Czujnik wtedy przechodzi w stan uśpienia i będzie w nim przebywał aż do pojawienia się następnego sygnału startu. Jeżeli sygnał na linii danych jest zawsze w stanie wysokim, oznacza to niepoprawne działanie czujnika, może to być spowodowane wadliwym podłączeniem.

## Algorytm odczytu pomiarów

Podczas odczytywania danych bardzo ważne są czasy występowania odpowiednich wartości logicznych. Błędna analiza czasu skutkuje złym zinterpretowaniem danych, co prowadzi do sfalszowania wyników.

Algorytm zastosowany przy tworzeniu projektu inżynierskiego jest bardzo prosty. Polega

on na odczytywaniu przy każdej możliwości stanu linii danych oraz zliczaniu ilości wystąpień stanu wysokiego, gdyż tylko ten koduje wysyłane wartości z czujnika.

Linia danych jest próbkowana do momentu odczytania 40 bitów. Jest ona ustawiana wówczas w stan wysoki, czujnik przechodzi w stan spoczynku. Po zakończeniu odczytu następuje analiza przetworzonych danych.

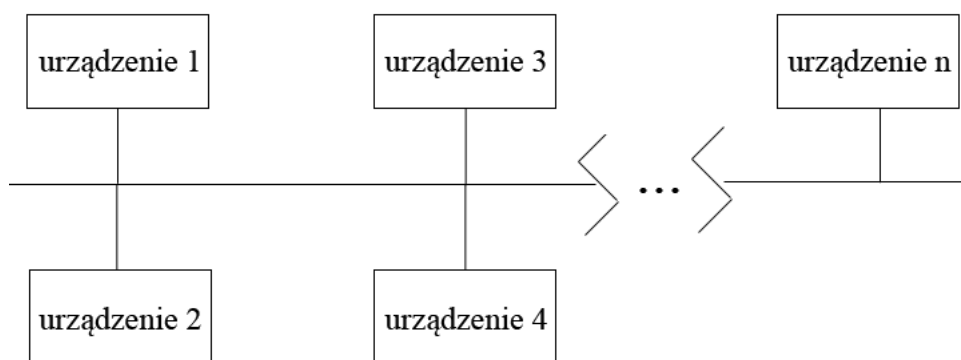
Ilości wystąpień każdego stanu wysokiego podlegają binaryzacji, czyli procesowi podziału na dwa zbiory. Zadaniem tej funkcjonalności jest znalezienie odpowiedniej wartości granicznej, takiej która jednoznacznie wyznacza ile razy musiał być spróbkowany sygnał na linii danych, aby został on zinterpretowany jako bit "1" lub "0". Zostało to zaimplementowane poprzez znalezienie minimalnej i maksymalnej wartości ilości wystąpień oraz wyliczeniu średniej tych dwóch liczb, wynik tego działania był wartością progową przy binaryzacji.

Po wyliczeniu progu binaryzacji, ilości wystąpień zostały poddane porównaniu z nią. Jeżeli ilość wystąpień była większa od progu, oznaczało to bit "1", w przeciwnym wypadku, było to kodowane jako "0".

## 6. Magistrale szeregowe

Magistrala jest to układ linii, po których przekazywane są wszystkie informacje pomiędzy podłączonymi do niej urządzeniami, np. komputerem, czujnikiem, regulatorem itp. Zasada działania magistrali opiera się na uzyskiwaniu oraz nadawaniu współpracującym częściom uprawnień do transmisji danych w danej jednostce czasu. W jednej chwili, w magistrali może działać tylko jedno urządzenie nadające oraz dowolna liczba odbiorców. Systemy o budowie opartej na magistrali są łatwo modyfikowalne oraz rozszerzalne, w prosty sposób można dołączyć lub odłączyć elementy systemu. Dane przesyłane na dużą odległość najlepiej jest przekazywać transmisją szeregową, na krótsze odległości, przesyłanie równoległe oraz szeregowe daje podobne rezultaty. Bity oraz całe słowa w tej komunikacji przesyłane są jeden po drugim. Przy takim sposobie łączenia się wystarczą tylko dwa przewody łączące odbiorcę z urządzeniem nadającym.

Przykład urządzenia w magistrali szeregowej został przedstawiony na poniższym schemacie:



Rysunek 6.1: Schemat magistrali szeregowej

Jak widać na załączonym rysunku, można podłączyć do magistrali wiele urządzeń. Wszystkie są podłączone do jednej linii danych, na której odbywa się komunikacja. To właśnie przez nią przesyłane są wszystkie dane pomiędzy elementami magistrali.



## 6.1. Magistrala I<sup>2</sup>C

Nazwa jest to akronimem od Inter-Integrated Circuit. Standard został opracowany w latach osiemdziesiątych przez firmę Philips.

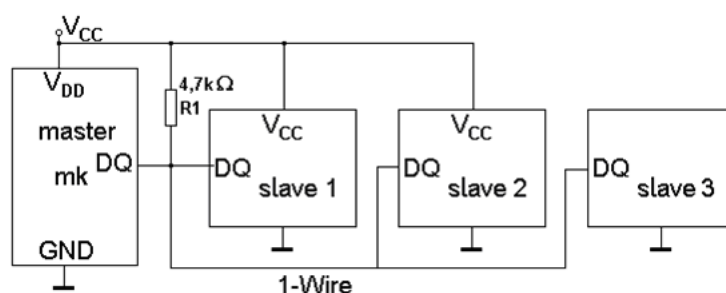
Jest ona bardzo często wykorzystywana w układach mikroprocesorowych, w sterownikach wyświetlaczy LCD, można ją stosować do sterowania pamięci RAM, EPROM, układami I/O.

Zaletami magistrali I<sup>2</sup>C są niewątpliwie takie właściwości jak: odporność na zakłócenia zewnętrzne, dodatkowe układy podłączone do niej mogą być dodawane lub wyłączane bez ingerencji w pozostały układ połączeń wcześniej stworzonych, połączenie na magistrali składają się tylko z dwóch przewodów, przez co ich ogólna liczba jest minimalizowana, wykrywanie błędów jest proste i łatwe do analizy, na magistrali może znajdować się wiele urządzeń typu master, umożliwiając kontrolę gotowych układów przez zewnętrzny komputer.

Magistrala I<sup>2</sup>C posiada dwie dwukierunkowe linie: dane są przesyłane przez Serial Data (SDA), natomiast sygnał zegara na Serial Clock (SCL).

## 6.2. Magistrala One-Wire

Magistrala 1-Wire jest to jednoprzewodowy interfejs szeregowy, który został opracowany przez firmę Dallas Semiconductors. W założeniach miał on umożliwiać łączność pomiędzy urządzeniami na małe odległości. Do magistrali tego typu również można podłączyć dowolnie wiele urządzeń elektronicznych. Jego największą zaletą jest fakt, że dane są wysyłane w obie strony, przy zastosowaniu tylko jednego przewodu, który również służy za zasilanie magistrali, dodatkowo wymagana jest osobna linia prowadząca do masy. Schemat podłączenia interfejsu One-Wire został zamieszczony poniżej:



Rysunek 6.2: Schemat magistrali 1-Wire

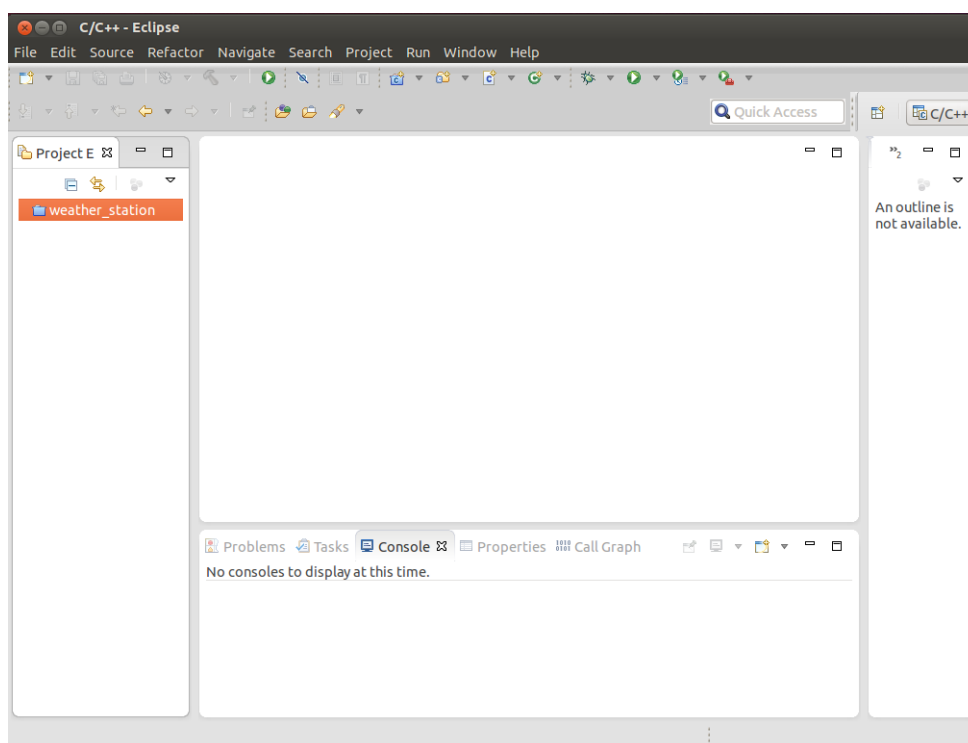
Magistrala, do poprawnego działania, wymaga rezystora podciągającego około 4,7 Ω do zasilania.

## 7. Programowanie mikrokontrolerów ARM

### 7.1. Używanie bibliotek Linux'a

### 7.2. Środowisko programowania

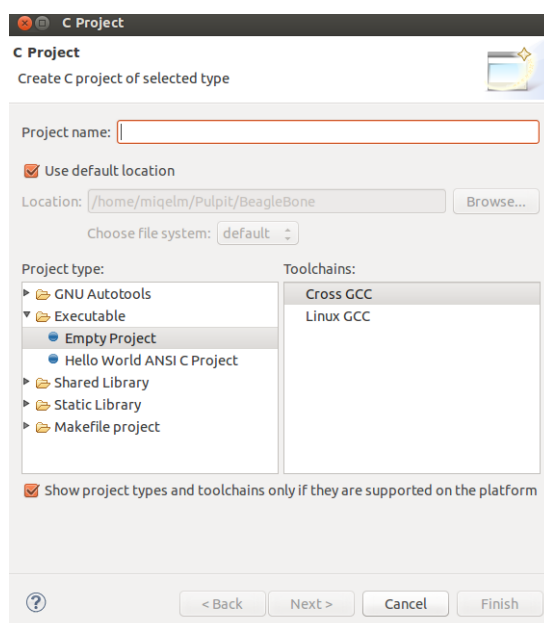
Kod programu oraz obsługi stacji pogody, zaimplementowanej na mikrokontrolerze Beagle-Bone black, został napisany przy użyciu środowiska programistycznego Eclipse. Środowisko to zostało wybrane przez wzgląd na ogromne możliwości, które ułatwiają w znacznej mierze programowanie oraz skracają jego czas. Eclipse jest darmowym narzędziem do programowania, jest intuicyjny w obsłudze, a przede wszystkim posiada wielką rzeszę użytkowników, przez co w przypadku problemów, ich rozwiązanie jest niemal natychmiastowe. Środowisko to można pobrać z oficjalnej strony, w projekcie został wykorzystany *Eclipse IDE for C/C++ Developers*.



Rysunek 7.1: Główny wygląd Eclipse'a

Po rozpakowaniu gotowego środowiska oraz jego uruchomieniu można już zacząć pracę z mikrokontrolerem, wystarczy jeszcze dokonać parę zabiegów, aby czas od zbudowania projektu, do jego uruchomienia na BeagleBone'ie była krótki. W celu uzyskania jak najwygodniejszej konfiguracji, został uruchomiony Eclipse na systemie operacyjnym Ubuntu, na którym były przechowywane wszystkie źródła. Dzięki odpowiedniemu dodatkowi do Eclipse'a - Remote System Explorer istnieje możliwość tworzenia programu na komputerze, jego cross-kompilacji do aplikacji wykonywalnej oraz uruchomienia gotowego pliku binarnego na mikrokontrolerze. Dzieje się to dzięki wspomnianemu wcześniej protokołowi SSH.

Aby stworzyć nowy projekt należy kliknąć File->New->C Project (może być również C++), pojawi się następujące okno:



Rysunek 7.2: Tworzenie nowego projektu

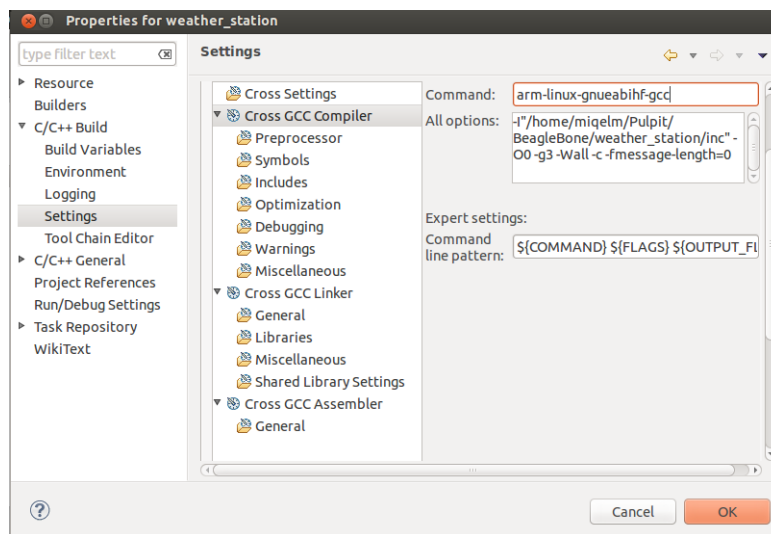
Po uzupełnieniu nazwy projektu oraz wyboru Cross GCC, przechodzimy dalej i kończymy konfigurację. Teraz następuje najważniejsza rzecz. Aby skompilować nasz projekt i móc go uruchomić na innej platformie sprzętowej, jaką jest procesor ARM, potrzebujemy cross-kompilatora. Jest to wymagane, gdyż komputer oraz BeagleBone różnią się budową oraz sposobem komunikacji na najniższym poziomie. W związku z tym, należy na komputerze zainstalować narzędzie umożliwiające nam generowanie pliku binarnego na inną platformę sprzętową, proces ten jest nazywany cross-kompilacją.

BeagleBone Black jest wyposażony w procesor o architekturze ARM hard float, dlatego też potrzebujemy do niego kompilatora, nazywa się on arm-linux-gnueabi-hf-gcc. Aby go zainstalować, należy w konsoli użytkownika wpisać następującą komendę:

```
sudo apt-get install arm-linux-gnueabi-hf-gcc
```

Potwierdzając chęć zainstalowania oraz pomyślnym przebiegu instalacji, jesteśmy w stanie teraz skompilować program na architekturę ARM.

W Eclipse klikając teraz prawym przyciskiem myszy na nowo stworzony projekcie, następnie wciśnięciu Properties, ukazują nam się właściwości projektu. Należy teraz zakomunikować środowisku, że program będzie kompilowany przy użyciu zainstalowanego przed chwilą kompilatora. W tym celu należy uruchomić zakładkę C/C++ Build, a potem opcję Settings i Cross GCC Compiler, w polu Command należy wpisać nazwę cross-kompilatora. Poniżej zostaje zamieszczony zrzut ekranu przedstawiający zaistniałą sytuację:



Rysunek 7.3: Ustawienia projektu

W podobny sposób należy wypełnić również pola Command w zakładkach Cross GCC Linker oraz Cross GCC Assembler, ten ostatni należy wypełnić wpisując: arm-linux-gnueabi-hf-as.

## 7.3. Kompilator

## **8. Tworzenie aplikacji zbierającej dane pomiarowe**

## **9. Przechowywanie oraz wyświetlanie wyników**

### **9.1. Serwer HTTP**

### **9.2. Baza danych MySQL**

### **9.3. Interfejs użytkownika**

## **10. Rezultaty pomiarów**

### **Dostosowywanie danych**

## **11. Podsumowanie**



# Bibliografia

- [1] Jacek Bogusz *Lokalne interfejsy szeregowo w systemach cyfrowych*. Wydawnictwo BTC, Warszawa, 2004.
- [2] Wojciech Mielczarek *Szeregowo interfejsy cyfrowe*. Wydawnictwo HELION, Gliwice, 1993.
- [3] Michael Leonard <http://www.michaelhleonard.com/cross-compile-for-beaglebone-black/>. [Dostęp: 11.12.2013].
- [4] BeagleBone Black Wiki <http://circuitco.com/support/index.php?title=BeagleBoneBlack>. [Dostęp: 11.12.2013].
- [5] Specyfikacja czujnika ciśnienia BMP085 <https://www.sparkfun.com/datasheets/Components/General/BMP085-DS000-05.pdf>. [Dostęp: 11.12.2013].
- [6] Specyfikacja czujnika ciśnienia DHT-22 <http://www.adafruit.com/datasheets/DHT22.pdf>. [Dostęp: 11.12.2013].

## **A. Kod źródłowy aplikacji**