MainBoard User Manual

A guide to using MainBoard middleware

v 1.1

UNIZG

September 28, 2017

Chapter 1

Connecting to aMussel over Bluetooth

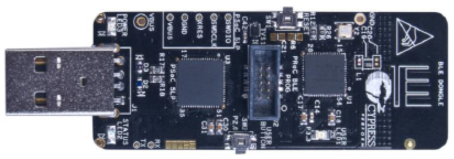


Figure 1.1: BLE USB dongle (CY5670)

Necessary hardware components:

- Bluetooth (BLE) module on MainBoard (MB) – each aMussel comes with an embedded BLE module
- BLE dongle that needs to be connected to user's PC USB port – dongle used for subCULTron
  The purpose of BLE dongle is to provide UART bridge between MB and PC. It's main purpose is to establish basic serial communication between MB on aMussel and PC.

In order to connect to aMussel over Bluetooth, the following prerequisites have to be met:

1. BLE dongle has to be programmed.

2. BLE module on MB has to be programmed with a certain aMussel ID.

1.1 Programming BLE dongle

The code for BLE dongle is provided in subCULTron's GitHub page (link), `BLE-Dongle` repository. To program BLE dongle you need to install PSoC Creator 4.0 IDE (here) and checkout the needed git repository.

Then:

1. Open the `BLE-Dongle` project in PSoC Creator. Build the project (right click on project name > Build `BLE_DONGLE`).

2. Set `BLE_DONGLE` project to active (right click on project name > Set As Active Project).

3. Connect BLE dongle to the PC's USB port

4. Program the dongle (*program* button in PSoC creator or Ctrl+F5).

5. If prompted, select a device to program.

1.2 Programming BLE module

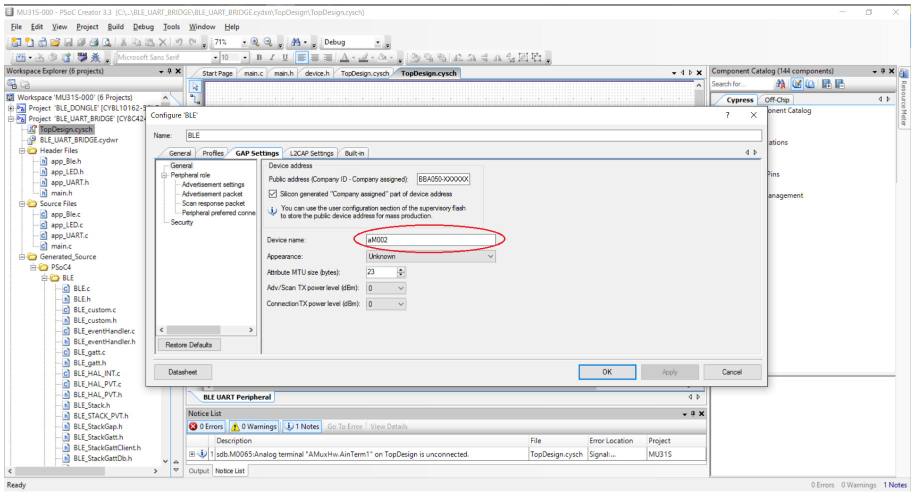
BLE module can be programmed using MiniProg programmer. The code is provided in `BLE_UART_BRIDGE` project of PSoC creator `MU31S-000` workspace, which can be found on project's GitHub page, `MainBoard-middleware` repository.

Steps:

1. Modify the project to define aMussel ID. To do so, open TopDesign.cysch and double click on BLE module. Go to GAP Settings > General and enter aMussel ID in Device Name field (see Figure 1.2). Naming convention is as follows: `'aMXXX'`, where `'XXX'` represents aMussel ID in range $[000 - 999]$, e.g. `'aM001', 'aM023'` ...

2. Build the project (right click on project name > Build `BLE_UART_BRIDGE`).

3. Set `BLE_UART_BRIDGE` project to active (right click on project name > Set As Active Project).



Configure 'BLE'

Name: BLE

Device address

Public address (Company ID - Company assigned): BBA050-XXXXXX

☑ Silicon generated "Company assigned" part of device address

You can use the user configuration section of the supervisory flash to store the public device address for mass production.

Device name: aM002

Appearance: Unknown

Attribute MTU size (bytes): 23

Adv/Scan TX power level (dBm): 0

Connection TX power level (dBm): 0

	Description	File	Error Location	Project
1	sdb.M0065:Analog terminal "AMuxHw.AinTerm1" on TopDesign is unconnected.	TopDesign.cysch	Signal:...	MU31S

Figure 1.2: Setting the BLE device name

4. Connect MiniProg to BLE module and the PC.

5. Program the device (*program* button in PSoC creator or Ctrl+F5).

6. If prompted, select a device to program.

1.3 Communicating with MU using BLE

To successfully send and receive data between aMussel and PC, we recommend the following setup:

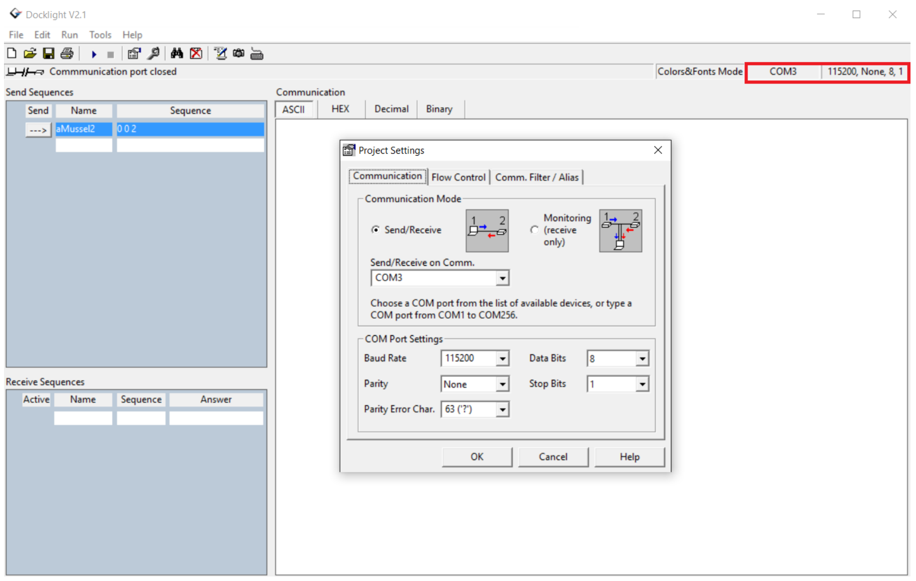
1. Connect the programmed BLE dongle to your PC.

2. Open Docklight [1] (or any other tool for serial communication).

3. Connect to the correct COM port. Port can be selected by double clicking on top right sections of status bar (indicated with red rectangle in Figure 1.3). After selecting the port, make sure the settings are matching the ones in the figure. Now you can simply press the play button to establish connection.

4. When connected to BLE dongle, you should see the output in your serial communication program as presented in Figure 1.4. If the output is not

[1] `docklight.de/`



Commmunication port closed

COM3 | 115200, None, 8, 1

Name	Sequence
aMussel2	0 0 2

Communication Mode

(● Send/Receive) Monitoring (receive only)

Send/Receive on Comm.: COM3

Choose a COM port from the list of available devices, or type a COM port from COM1 to COM256.

COM Port Settings

Baud Rate	115200	Data Bits	8
Parity	None	Stop Bits	1
Parity Error Char.	63 ('?')		

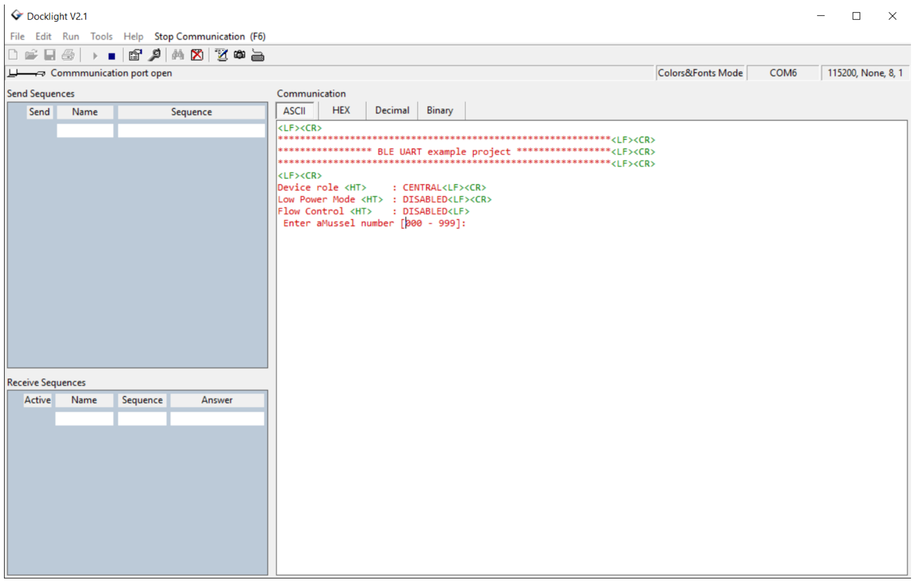
Figure 1.3: Selecting the COM port

there or the program is stuck for any reason, you can always reset it by pressing the user button on dongle (see Figure 1.5).

5. The next step is to select device ID you wish to connect to. You do so by sending a sequence representing device's unique identifier. For example, if BLE board on aMussel is programmed to have ID `001`, the sequence needs to match it. For reference, look at Figure 1.6.

6. Upon connection establishment the program prints out:
```
Server with matching custom service discovered...
Connection established
Notifications enabled
Start entering data:
```

7. Now the connection is established and you can start sending and receiving data.



```
<LF><CR>
************************************************************<LF><CR>
***************** BLE UART example project *****************<LF><CR>
************************************************************<LF><CR>
<LF><CR>
Device role <HT>     : CENTRAL<LF><CR>
Low Power Mode <HT>  : DISABLED<LF><CR>
Flow Control <HT>    : DISABLED<LF>
 Enter aMussel number [000 - 999]:
```

Figure 1.4: BLE bridge status

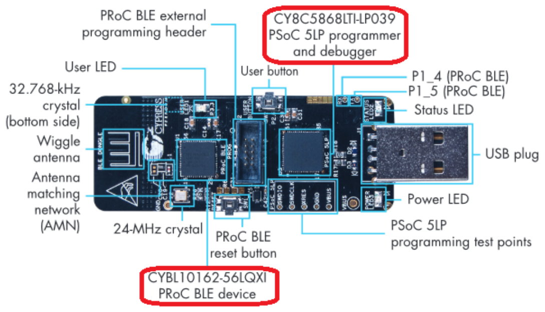
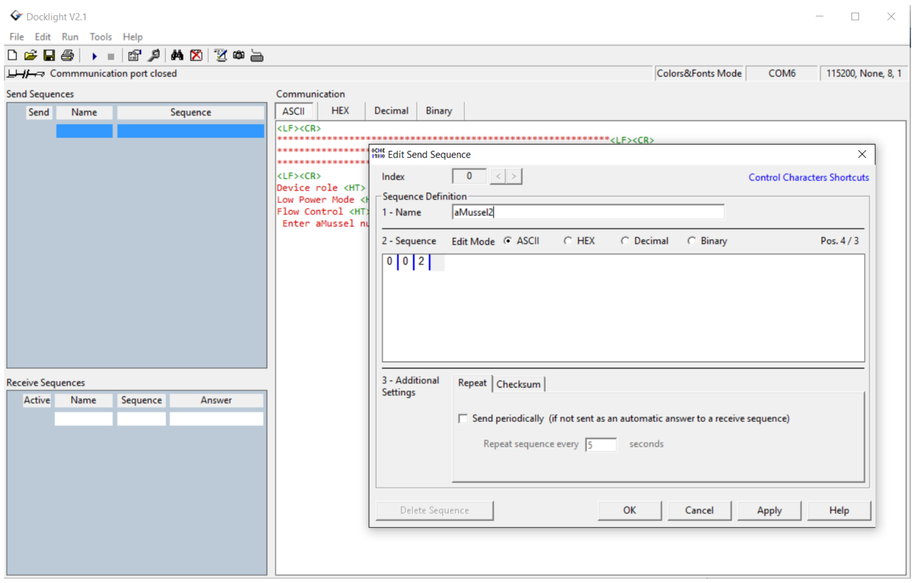

PRoC BLE external programming header
CY8C5868LTI-LP039 PSoC 5LP programmer and debugger
User LED
User button
P1_4 (PRoC BLE)
P1_5 (PRoC BLE)
32.768-kHz crystal (bottom side)
Status LED
Wiggle antenna
USB plug
Antenna matching network (AMN)
Power LED
24-MHz crystal
PRoC BLE reset button
PSoC 5LP programming test points
CYBL10162-56LQXI PRoC BLE device

Figure 1.5: CY5670 USB Dongle



Commmunication port closed | Colors&Fonts Mode | COM6 | 115200, None, 8, 1

```
<LF><CR>
****************************************************************<LF><CR>
****************
****************
<LF><CR>
Device role <HT>
Low Power Mode <H
Flow Control <HT>
 Enter aMussel nu
```

Index: 0

Sequence Definition

1 - Name: aMussel2

2 - Sequence — Edit Mode: [✓ ASCII] HEX Decimal Binary — Pos. 4 / 3

```
0 0 2
```

3 - Additional Settings — Repeat | Checksum

☐ Send periodically (if not sent as an automatic answer to a receive sequence)

Repeat sequence every 5 seconds

Figure 1.6: Selecting a device to connect to

Chapter 2

Programming MainBoard

2.1 Programming using MiniProg (wired)

Add figures and text.

2.2 Programming using RaspberryPI and I2C Bootloader (wireless)

Bootloader is a short program used to burn the firmware to the microcontroller without any programmer device. A prerequisite to using Bootloader to program the Main Board is having the Bootloader itself configured and programmed onto the board.

In order to program Main Board using Bootloader, `Bootloadable` component on MB has to be running and awaiting for programming request. Main Board has always a *interpreter* task running which activates Bootloadable on request. This mode is entered by sending `'b'` character to MB over BLE.

For programming over Raspberry PI we use `I2C_BOOTLOADER` that is given as a project of PSoC creator `MU31S-000` workspace. In order to program I2C bootloader to MainBoard, a MiniProg wired programmer is necessary.

Steps:

1. Build the bootloader project

2. Set `I2C_BOOTLOADER` project to active (right click on project name > Set As Active Project).
3. Connect MiniProg to MB and the PC.
4. Program the MB (*program* button in PSoC creator or Ctrl+F5).
5. If asked, select a device to program. Sometimes the connection between MB and MiniProg is bad so make sure you adjust the cable until the correct device is detected.
6. Once the device is detected, select `Port Acquire` > `Connect` > `OK` and wait for the completion of programming process.

The previous procedure should be done only once in a lifetime for each MainBoard.

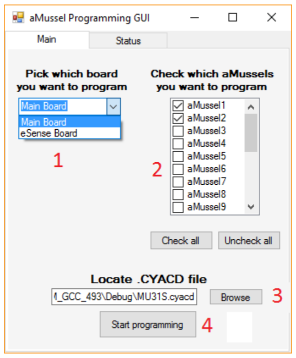
The programming procedure starts by transferring the desired program to aMussel's Raspberry Pi over WiFi. Then the program compiles on Raspberry Pi. Programming is done over I2C, where Raspberry Pi reprograms the Main Board.

Steps:

1. Build the project you wish to program in PSoC Creator.
2. Put Main Board in Bootloader mode using BLE. aMussel then starts blinking blue/green light.
3. Make sure Access Point router is turned on (and setup properly) and your PC is on the same network as aMussel.
4. Wait for aMussel to boot up the Raspberry Pi
5. Open aMussel programming GUI and follow programming steps described in 2.2.1. aMussel blinks red light when it is being programmed.
6. Wait for programming to finish. When blinking stops, the aMussel is programmed.

2.2.1 Windows programming application

To make the process of reprogramming robotic swarm easy, an automated programming application was made. The application was made in Visual Studio, using the Visual C++ programming language.



Pick which board you want to program

Main Board

Main Board
eSense Board

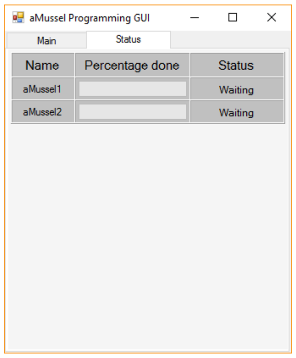
Check which aMussels you want to program

- [x] aMussel1
- [x] aMussel2
- [ ] aMussel3
- [ ] aMussel4
- [ ] aMussel5
- [ ] aMussel6
- [ ] aMussel7
- [ ] aMussel8
- [ ] aMussel9

Locate .CYACD file

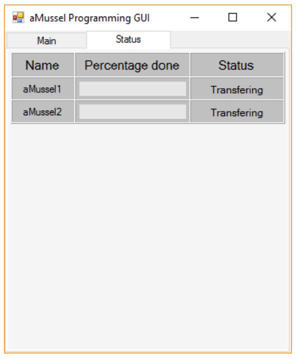
_GCC_493\Debug\MU31S.cyacd

(a) Main screen



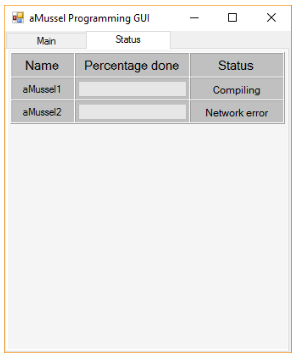
Name	Percentage done	Status
aMussel1		Waiting
aMussel2		Waiting

(b) Status screen - Idle



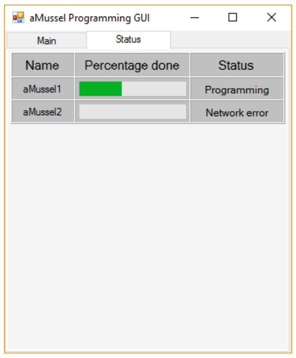
Name	Percentage done	Status
aMussel1		Transfering
aMussel2		Transfering

(c) Status screen - Transfering code



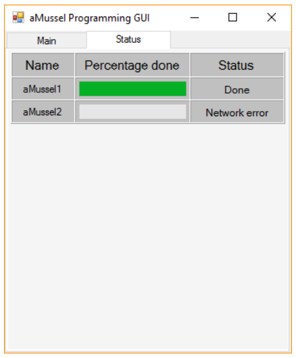
Name	Percentage done	Status
aMussel1		Compiling
aMussel2		Network error

(d) Status screen - Compiling code



Name	Percentage done	Status
aMussel1	🟩	Programming
aMussel2		Network error

(e) Status screen - Programming code



Name	Percentage done	Status
aMussel1	🟩	Done
aMussel2		Network error

(f) Status screen - Programming done

Figure 2.1: Windows programming application

The programming procedure is divided in five steps:

- Choosing a programming configuration
- Starting the programming procedure
- Transferring the binary file containing the code
- Compiling the program on RaspberryPI for programming the PSoC board over I2C
- Executing the compiled program

As shown in 2.1a), choosing a programming configuration is split in three steps:

- Choosing the appropriate board
- Choosing which aMussels the operator wants to program
- Choosing the binary file containing the code

Before the pressing *Start programming* button in step 2., in the status tab aMussels 1 and 2 are shown to be in waiting mode (2.1b)). After the programming procedure has started, the status indicator of each aMussel is indicating that the transferring procedure has started (2.1c)). In case of an unsuccessful file transfer, the status indicator shows "Network error", as seen in aMussel2's status(2.1d)). The compiling step is shown in 2.1d in aMussel1's status. In the final stage of the programming procedure, the progress bar indicates the percentage of flash memory currently transferred to the PSoC board (2.1e)). After successful programming, the status indicator of aMussel1 is in the "Done" state and the aMussel should start executing the newly programmed code.

Chapter 3

Testing device functionality

For individual device functionality test a special task which interprets commands sent through serial port and calls corresponding API functions has been devised. It is called `test_function` and is located in `main.c` of `MU31S` project.

To send commands, a serial connection to MB's general purpose UART needs to be established. All of the commands and debugging messages are then sent through that link.

To test the functionality, a special sequence of messages has been defined. The controls are found in the table 3.1.

Table 3.1: MB testing commands

Bootloader		
b		enter Bootloader
Power board		
P	1	switch to main battery
P	2	switch to backup battery
P	3	disable charging
P	4	get main battery voltage
P	5	get backup battery voltage
P	6	get main battery current
P	7	get backup battery current

P	8		get supply current
P	9		get main battery's charging status
P	A		get backup battery's charging status
P	B		enable charging
Electric sense			
e	b		esense board program
e	0		esense board init
e	s		esense board 's' command
e	m		esense board 'm' command
e	+		esense board '+' command
e	-		esense board '-' command
e	a		esense board get angle
e	d		esense board get distance
Turbidity sensor			
t			Get turbidity sensor reading.
GSM/GPS module			
g	c		call a number defined in test function
g	h		hang up
g	s		send SMS to a number defined in test function
g	m		receive SMS
g	u		list all unread SMS
g	a		list all SMS
g	l		clean read SMS
g	N		power on GPS module
g	F		power off GPS module
g	f		get GPS fix status
g	k		get GPS data
SPI IMU			
p	a		get accelerometer data
p	g		get gyroscope data
p	m		get magnetometer data
p	c		calibrate imu
LEDs			
l	r	num	turn red with intensity *num*
l	g	num	turn green with intensity *num*
l	b	num	turn blue with intensity *num*

l	o		turn off
l	l	num	turn into rgb defined by *num*
Buoyancy motors test			
m	u		go up
m	d		go down
m	s		stop
Pressure and temperature sensor			
r	r		reset
r	p		get pressure
r	t		get temperature
Scenarios			
s	x		power Pi on (with faulty scenario on purpose)
s	C		turn off Pi (following the procedure)
s	Z		turn off Pi (on pin)
s	1		experiment 1
s	2		experiment 2
s	3		experiment 3
s	4		experiment 4
...			
x			Cancel currently running scenario.