Swarmbee - Xmega communication guide
Project swarming

E. VAN SPLUNTER, M. VISSER AND M. SIEKERMAN
Amsterdam university of applied sciences, TU Delft
June 27, 2016

Abstract

This manual will give an overview of how to use Nanotron's Swarmbee module in combination with the Atmel Xmega128A4U microcontroller. This manual is intended for the students who intend to further develop this project. To fully understand this manual, knowledge about the Xmega microcontroller and the purpose of the swarming module and swarming in general is required.

Swarmbee API

The swarm bee module works with a set of API commands which can be found in the "Swarm API" document from Nanotron. When a certain command was been send to the swarm bee it will reply with the requested value. For instance, when we want the node ID, we send:

```
1 GNID
```

The reply will be:

```
1 =001122334455
```

All commands which return one single line, the reply begins with '='. All commands which return multiple lines, the reply begins with '#'. All asynchronous lines, such as broadcast messages start with '*'. For this implementation it is very importand that the module ID's and the ranges between them are obtained, this van be done by a ranging request broadcast.This broadcast message can be requested at a certain interval which depends on the demand of this information. The broadcast interval can be set via a command; "sbiv 1000", where sbiv stands for "set broadcast interval" and the time is set at 1000ms, or 1 second. When the swarmbee receives a ranging request its response will look somewhat like this:

```
1 *RRN :1 F3123123133 ,1 F3CFF322133 ,0 ,001843
```

Where the first two hexadecimal values represent the source ID (sending module), and the destination ID (receiving module). This is followed up by an error code and the range relative in centimetres to the destination node.

The commands from the API list can be tested in the "swarm-pc-tool" from Nanotron.

Hardware

Swarmbee module

The following drawing shows the PCB placement of the Swarmbee module. The detailed function of connectors X2 is described in figure 2. For more information consult Nanotron's documentation.

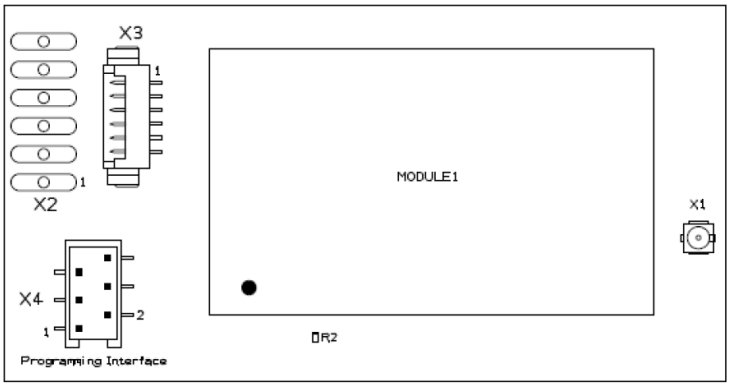

Figure 1: overview of the Swarmbee module

X2/1	MOD_EN	*swarm* bee Module Enable
X2/2	GND	Ground
X2/3	VIN	Input voltage + 3V … +5,5 V
X2/4	UART_RX	Incoming signal to *swarm* bee
X2/5	UART_TX	Outgoing signal from *swarm* bee
X2/6	GND	Ground

Figure 2: Pinout table of the Swarmbee module

Xmega - Swarmbee

The Xmega sends request and receive data from the Swarmbee module. This information is transmitted using the UART protocol. The communication is mainly used for sending API commands to the Swarmbee module. The UART connecting uses a baudrate of 115200 bps since this is the transmission speed of the Swarmbee module. The Xmega uses two pins, RX (PC2), TX (PC3) to achieve a wired connection with the Swarmbee. For debugging pin PC7 en PC8 are reserved. The debugging connecting can be established with a terminal such as Putty.

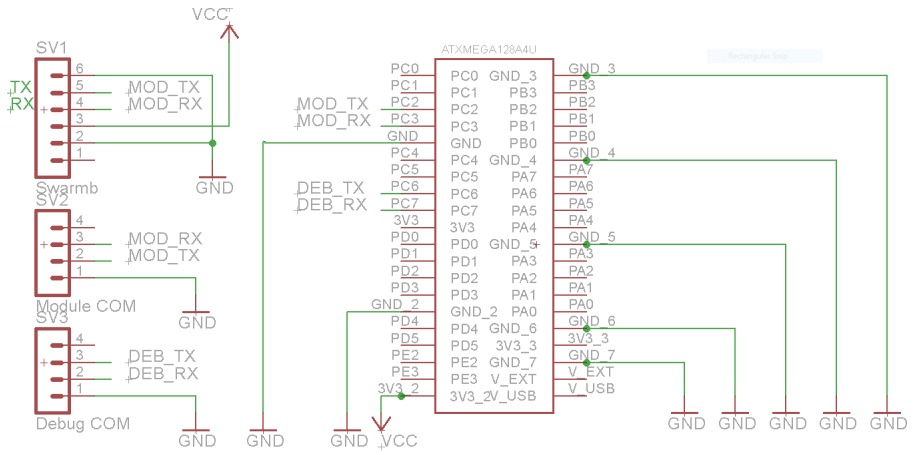

Figure 3: Wiring diagram of the xmega and the Swarmbee module

Figure 3 shows the wiring diagram made for a PCB to connect the Xmega with the Swarmbee and to easily connect the two serial ports.

Software

In the software we first declare the necessary things like UART initialization and clock speed. To communicate with the swarm bee we use: "command();", to print something to the debug log we use "DebugPrint();". In main.c, the custom message of the ranging request notification is set to 0 because we don't need any extra information at this time, this can be set to your own needs. Also the interval of the broadcast message gets set, these are represented by the following lines:

```
1 Command(NCFG0);
2 Command(SBIV500);
```

In the while of main.c, the function "DetermineCommandtype();" gets executed when a ranging request notification has been received, this is checked by the function: ValidateMessage(),. If a message is corrupted this will not get trough.

```
1 if (ValidateMessage(message,TYPE_RRN) == true){
2   DetermineCommandtype(message);
```

This function can be found in transreceive.c, This is where the incoming messages get decomposed and analysed, character for character. The beginning of each message will be compared to an expected order of characters, this can be extended with any kind of expected message. When the received string starts with "*RRN", we know that the string can be used be used to fill the population list, function: "fillpopulationlist();" will be executed.

```
1 if     (strcmp(command, "*RRN") == 0){
2   fillpopulationlist(messagePointer);
```

In the function "fillpopulationlist.c" the information from the received notification get seperated by the commas between the values. this function uses the functions from list.c to fill the dynamic array. The following functions are included in list.c:

- PrintHeaderList(); This function prints the static information of the list.
- print_list(); This funtion prints the dynamic information of the list.
- append(); The append function fills the list from the end of the list.
- insert(); The insert function fills the list from the beginning of the list.
- popListByNumber(); Finds and replaces data by the index numbers of the list.
- popListByValue(); Finds and replaces data by the list variable to be found.
- sizeOfList(); Returns the size of the total list.

When a ranging request of 5 nodes has filled the dynamic array, the populationlist will look like follow:

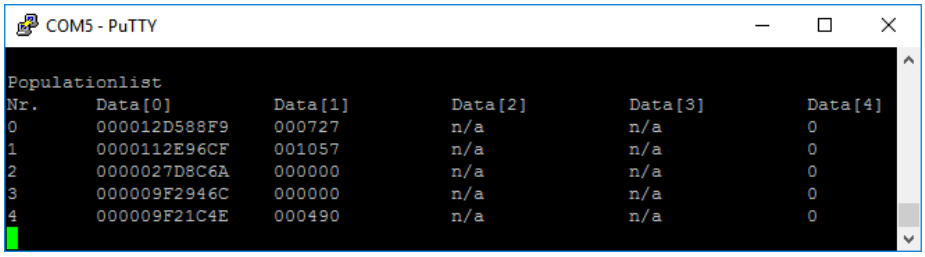

Figure 4: Putty log of the filled population list