1. Register in arm Mbed site in order to use their online compiler.
2. When inside Mbed compiler choose the propriate Mbed board used for the PSI swarm robot (mbed LPC1768)
3. Import the PSI swarm robot library into the compiler from the developers mbed repository ( <https://os.mbed.com/teams/Psi-Swarm-Robot/>). Choose the latest version at the time which is v9
4. When the library is loaded int the compiler, navigate the PSISwarmV9 folder and open the serial.cpp file.
5. Find the “SerialControl::IF\_pc\_rx\_callback();” function and change it as follow

void SerialControl::IF\_pc\_rx\_callback()

{

int count = 0;

char message\_array[255];

pc\_command\_timeout.detach();

while(pc.readable()) {

char tc = pc.getc();

message\_array[count] = tc;

wait(0.001);

//printf("input: %c and bite number: %i\n",message\_array[count],count); //testing

count ++;

message\_array[count]=0;

/\*if(pc\_command\_message\_started == 1) {

if(pc\_command\_message\_byte == 3) {

pc\_command\_timeout.detach();

if(tc == COMMAND\_MESSAGE\_BYTE) {

// A complete command message succesfully received, call handler

pc\_command\_message\_started = 0;

count = 0;

IF\_handle\_command\_serial\_message(pc\_command\_message , 0);

} else {

// Message is not a valid command message as 5th byte is not correct; treat whole message as a user message

pc\_command\_message\_started = 0;

message\_array[0] = COMMAND\_MESSAGE\_BYTE;

message\_array[1] = pc\_command\_message[0];

message\_array[2] = pc\_command\_message[1];

message\_array[3] = pc\_command\_message[2];

message\_array[4] = tc;

count = 5;

}

} else {

pc\_command\_message[pc\_command\_message\_byte] = tc;

pc\_command\_message\_byte ++;

}

} else {

if(count == 1) {

if(tc == COMMAND\_MESSAGE\_BYTE) {

pc\_command\_timeout.attach(this,&SerialControl::IF\_pc\_rx\_command\_timeout,command\_timeout\_period);

pc\_command\_message\_started = 1;

pc\_command\_message\_byte = 0;

}

}

}\*/

//printf("total input : %s\n",message\_array); //testing

}

if(/\*!pc\_command\_message\_started && count>0\*/1) IF\_handle\_user\_serial\_message(message\_array, count, 0);

}

Explanation: This function is handling PC serial messages and a command system that is not documented by its developers and we are not using them. In order to avoid the messages received from the computer to be split or delayed, we are commenting this section out in a way to send the received message back to the main loop to be used by our program.

6. In the main.cpp part of the library creates a global string named command to hold the command and pass it to the main loop

7. in the “handle\_user\_serial\_message” function that receives and handles the user PC serial messages, we need to pass it to our global variable command

command = message;

8. Now that we can use the message that is received from the computer as a string, we can use it in the main loop to call the appropriate functions required by the program running on the PC device. We do this using IF statement to check if the user message is a recognized function from a list and execute it on conformation. This list of messages is meant to communicate the robot API with the python/Ros API internally without user interference therefore is not included.

A ready version of this code can be found on the got hub repository ([git@github.com:LJamieson/Robot.git](mailto:git@github.com:LJamieson/Robot.git)) under the name of [v4\_gnuarmeclipse\_lpc1768](https://github.com/LJamieson/Robot/blob/master/Robot_V3_gnuarmeclipse_lpc1768%20(1).zip).zip.

9. Now that we have the code ready, we need to compile it using the compile button in the mbed compiler and save the result .bin file at any assessable location on the local storage.

The binary file can also be found in the same Git repository under the name of V4Final.LPC1768.bin

10. Next step is uploading the .bin file to the Mbed board on the robot by connecting it to the devise using a micro usb to usb cable.

The Mbed board will appear as an external drive location when plugged and allow the binary filed to be copied and placed on it.

11. Pressing the small button on the mbed board will restart it and load the newest version of the binary file placed in its memory.

12. Now we can send commands to the robot using serial interface, for example terminal or python script etc. The settings used to communicate with the mbed board are 115200 Bound ride with 8 bit of data followed by 1 stop bit.