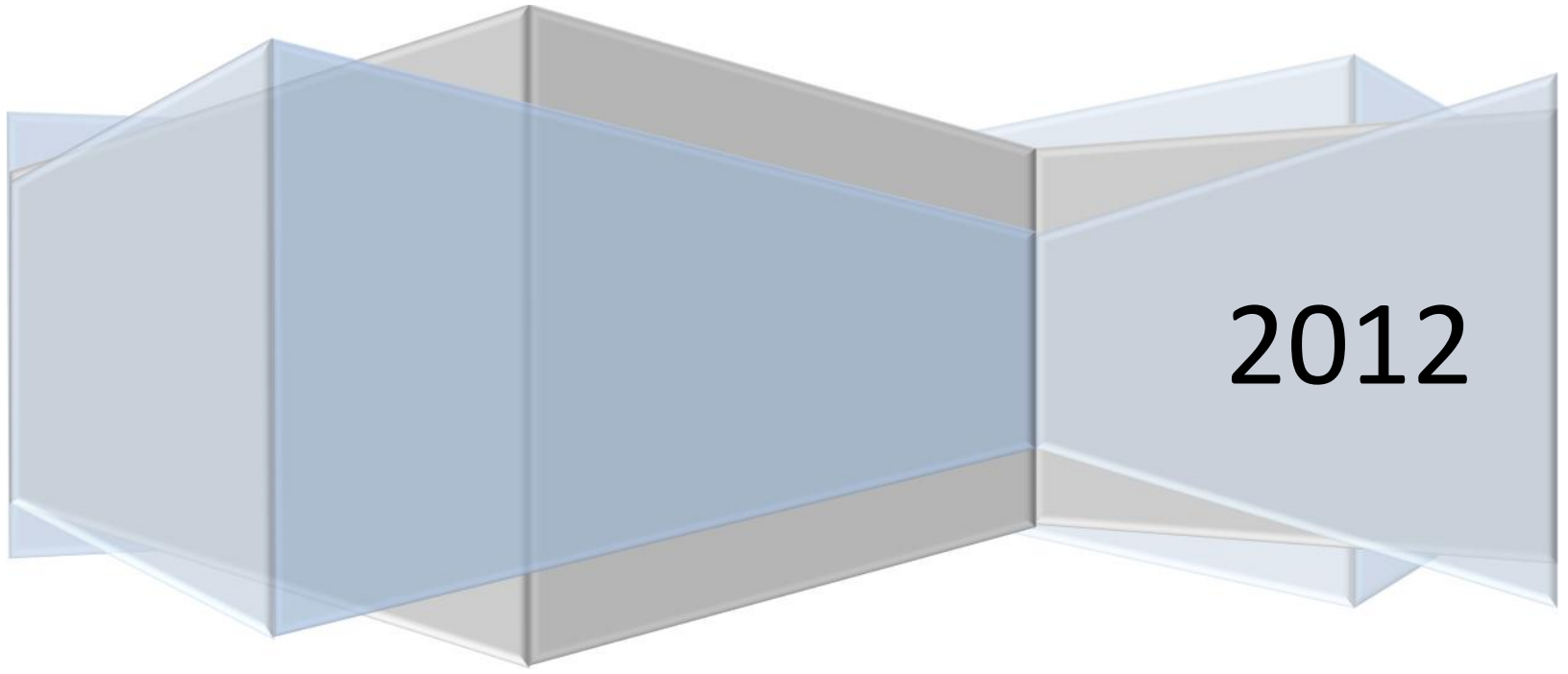
Montgomery County Community College

Preparing and flying the airborne mission computer –R9 Final

Research Document – G06

EGR299 Students and Engineering Faculty



2012

Table of Contents

THIS PAGE BLANK

Introduction

openPicus[1] is an Italian company producing hardware for the Internet of Things. The modules operate on a powerful open source operating system, FreeRTOS[2], and are embed with a full dedicated TCP/IP software stack as well as web server. The software stack, as well as the IDE, is released as open source software.

The company produces mostly two hardware products:

- Flyport[3] is a system on module based on Microchip Technology[4] PIC processors with different kind of connectivity to Internet: Wi-Fi and Ethernet. Flyport modules are used to connect and control systems over the Internet through an embedded customizable webserver or the standard TCP/IP services. Flyport modules come with microcontroller and transceiver all in one. The microcontroller runs your applications, no host processor needed. Pinout is customizable by software.
- Nest expansion boards that are created for specific applications, compatible with each Flyport.
- Also a free IDE development tool is provided by the manufacturer to create, compile and download firmware applications to the modules and for importing external web pages.

The complex communication stack that works in the background managed by FreeRTOS helps control the events. Flyport modules are based on the Microchip Technology PIC microcontrollers and a communication Transceiver, Wi-Fi or Ethernet. Different modules, such as Flyport Wi-Fi and Flyport Ethernet, are mechanically and pin to pin compatible with each other in order to save on expansion board and give greater customizability.

The product has also been interfaced with several cloud servers such as Paraimpu or ThingSpeak. The magazine Postscapes has placed the openPicus team into the list of the 100 most influential Internet of Things thinkers.

This document consists of 5 chapters:

- The Flyport modules FAQ
- The openPicus IDE
- Programming and downloading a project
- Flying a project on the onboard mission computer
- The FileZilla FTP server

[1] http://www.openpicus.com/
[2] http://en.wikipedia.org/wiki/FreeRTOS
[3] http://www.openpicus.com/site/FlyportModule
[4] http://www.microchip.com/

The MC3 students and faculty recognize and show appreciation to Ing. Claudio Carnevali and Ing. Gabriele Allegria from openPicus in Rome, Italy for our academic tie-up. Special thanks to Mr. Dan Graboi from FlyportAmerica, in California, USA for helping us in developing our airborne mission computer.

Credits

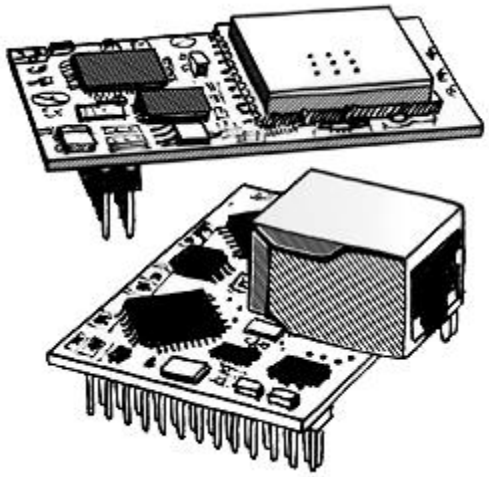
The EGR299 class students

Aaron Wilz
Brian Jerardi
Daniel Rowe
Jacob Simon
Jason Cassel
Kevin Healy
Michael Marino
Michelle Gavan
Scot Kantner
William Fellmeth

The engineering faculty

Andrew Ippolito
Bill Brownlowe
Jean-Jacques Reymond

Chapter 1 – The Flyport modules FAQ



The Flyport is a "system-on-module" solution based on a PIC 16-bit processor and a Communication Transceiver. They are two Pin-to-Pin compatible versions: Flyport Wi-Fi and Flyport Ethernet. A GPRS module is coming soon.

The openPicus software framework that empowers each Flyport is based on:

- FreeRTOS: an open source, ultra efficient and ultra-reliable operating system that also manages the communication protocol stack.
- Internet Connectivity: TCP and UDP, a customizable web server, E-mail and FTP client.

Flyport applications are developed by leveraging C and C++ development skills, thanks to the powerful and free IDE.

FAQ

What is the Flyport?

Flyport is a programmable and open source "system-on-module" technology, based on the Microchip PIC processors with different kinds of connectivity to Internet: Wi-Fi and Ethernet. A GPRS module is coming soon.

Flyport modules are an easy way to control and connect systems over the Internet. This way, home automation becomes much easier and efficient. With Flyport you can control your devices though a browser, easily on your computer, a tablet and/or a smartphone. The Flyport modules also offer an integrated, programmable dynamic webpage engine.

What kind of software is the Flyport hardware using?

The Flyport internal software is based on FreeRTOS that manages the communication protocol stack, no need for communication expertize.

How do I program a Flyport module?

Flyport comes with a free IDE environment for programming the modules. The language used to program the modules is C/C++-like. Flyport modules are based on PIC microcontrollers. The developers just need C/C++ expertise to create the application. The Flyport manages all the required network interfacing, internet communications protocols and the internal Webserver.

What do I need to program and use a Flyport module?

To get started with the Flyport you will need:

- A Flyport module, either the Flyport Ethernet or the Flyport WiFi.
- A USB-to-Serial programmer module.
- The free Flyport IDE Software.
- A Microsoft Windows computer or tablet, or a computer capable of emulating a Microsoft Windows environment.

What is so cool about the Flyport technology?

Flyport is a great technology that comes with some very important features:

- It is designed both for makers, hobbyists and prototype developers, as well as for those who need a networking embedded platform integrated on their product.
- It is low cost.
- It is an open platform.
- It has everything you need to get started with your projects: You don't need to worry about connectivity, external libraries, communication protocols and thread programming. It is all there.
- The GPIOs are more than enough even for your most demanding project, plus the fact that the pins are remappable at runtime level.
- It comes with great accessories that make your development easier and better. Extensions are designed and provided all the time.
- It has a great community of makers and users that contribute tutorials, libraries and knowledge.

Can I use Flyport technology for building Internet of Things (IoT)?

Flyport features exactly what an IoT project needs:

- Connectivity wired or wireless.
- Embedded web server.
- Ad-hoc and Infrastructure WiFi mode.
- Small power consumption.
- Comes with accessories that allow direct connectivity to sensors and actuators.
- Small size and lightweight.

What do I need to install the Flyport IDE?

A Windows computer or a Linux/MacOSX computer capable of emulating a Windows environment.

Where is the software to download?

Anything you need, from the Flyport IDE to external libraries, is located at the openPicus website[5].

Are there any Flyport IDE requirements/prerequisites?

You need to install first the C30 Microchip compiler[6] v3.24 Lite.

[5] http://www.openpicus.com/site/technology/downloads
[6] ww1.microchip.com/downloads/en/DeviceDoc/MPLABC30Combo_v3_24Lite.zip

Is the IDE open source? When can I get the sources?

Yes, you may find the recent sources in the source code repository[7].

What can I do with a Flyport Module?

You can add analog and digital sensors, collect information such as temperature, humidity, light conditions, radiation, air quality, etc. You can also connect actuators like relay switches and servo motors and build automation systems, systems that can control electrical appliances, small robotic systems, robotic cars, etc.

Where can I purchase Flyport modules and accessories?

You can visit the OpenPicus online store[8] and purchase directly or visit one of the retailers[9] near you,

Flyport Ethernet or Flyport WiFi?

It really depends on your application/project needs. If you need to build a system that will be installed on a permanent location and has wired Internet connection available then Flyport Ethernet is the right choice for you. If wired connectivity is unavailable or you want to have a tablet to be connected ad hoc and wirelessly, then WiFi is a better choice.

How many I/O are available?

Flyport modules have 26 digital pins. On the Webserver default configuration there are only 5 digital inputs and 5 digital outputs. This is just a "startup" configuration, and it can be changed at any time. Any of the 26 pin can be configured as digital input or digital output. There are also 4 analog pins available.

What is the Flyport Bootloader?

The bootloader is a small program that starts when the microcontroller boots and listens on the serial port for a special message. When it receives this special message, usually a string, it "understands" that the IDE wants to program the microcontroller, it reads the commands arriving on the serial port and writes them on the microcontroller memory using an RTSP, a real time serial programming technique. The Flyport uses an internal bootloader to program the microcontroller with just a serial connection, for example the USB-to-Serial connection in the USB nest, or just a UART connection.

Is it possible to reprogram the Flyport fieldly?

It is possible to reprogram the Flyport Ethernet, and soon the WiFi, using an FTP server.

[7] http://code.google.com/p/openpicus/
[8] http://store.openpicus.com/
[9] http://www.openpicus.com/site/buy/retailers

Is there a power-saving mode for Flyport WiFi?

Actually there are two power saving modes available: Hibernate mode and Sleep mode. Check the Flyport WiFi Energy Saving Mode[10] for more information and code samples.

Can I drive a step motor with a Flyport?

Flyport supports PWM and can be used for driving step motors. You can use your Flyport WiFi module to build your own robot.

Licensing and Intellectual property

Our technology is based on the Open Source model as we believe that there's no more room for closed innovation today. You are free to create your commercial products: this is perfectly in line with our licensing. OpenPicus supports the Open Source Hardware Association[11] efforts and strives for 100% compliance with the initiative definitions. Here you can find the most common questions about Licensing and Intellectual Property.

Can I use openPicus or Flyport brands?

No, you cannot. Those are the only things that you can't use in your product: in few words, the naming of your products cannot include neither openPicus nor Flyport. Some examples:

- *MyOpenPicus Board = NOK*
- *MyPicus = OK*
- *Flyport Car = NOK*
- *Catport = OK*
- *RoboFlyport = NOK*

Can I use openPicus hardware schematics to create my own product?

Yes you can, even for commercial purposes: you only need to give a clear attribution to openPicus when releasing the product. A good attribution is "Powered by openPicus technology". Our schematics are released under CC-BY 3.0[12].

[10] http://wiki.openpicus.com/index.php?title=Flyport_WiFi_Energy_Saving_Mode
[11] http://www.oshwa.org/
[12] http://creativecommons.org/licenses/by/3.0/

Can I use openPicus Software to create my own product?

Yes you can, but please note that:

- FreeRTOS has its own license[13].
- The openPicus Framework (eg: tcp/ip stack, email sender, FTP, etc...) is available under LGPL v3.0[14].
- The Code Samples, Applications, Projects and Libraries are available in Apache License 2.0[15].

In a few words: you release software that links and uses unaltered code from openPicus, both Framework or Code Samples, App notes, etc. without any implication on your license choice. If you tweak the framework code you shall release it under an LGPLv3 compliant license.

Can I modify the IDE code

The IDE Software is in the process of being released under GPLv3. We'll be adding the code soon on a code repository.

[13] http://www.freertos.org/a00114.html
[14] http://www.gnu.org/licenses/lgpl.html
[15] http://www.apache.org/licenses/LICENSE-2.0.html

Chapter 2 – The openPicus IDE

To get started, you need a Windows PC or tablet with a USB port and Microsoft .NET Framework 4 is required to develop applications with the openPicus IDE.

1. http://www.microsoft.com/en-us/download/details.aspx?id=17851

The .NET Framework 4 web installer package downloads and installs the .NET Framework components required to run on the target machine architecture and operating system. An Internet connection is required during the installation.

In addition, we will be required to download and install the following files:

2. The C30 compiler free version from Microchip Technology, Inc.

   http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en010065

3. The IDE 2.2 from openPicus

   openpicus.googlecode.com/files/OpenPicus%20-%20Flyport%20v2.2.zip

Finally, download and read the following documents from openPicus:

4. The IDE 2.2 user manual

   http://openpicus.googlecode.com/files/IDE%202.2%20-%20Flyport%20user%20guide%20-%20rev1.0.pdf

5. The Flyport programmer's guide

   http://openpicus.googlecode.com/files/FLYPORT%20Programmer%27s%20Guide%202.2%20release%201.0.pdf

You are now ready to connect the miniUSB programmer to your PC or tablet, a serial bootloader is already on the module. Run the IDE and create a new project or download an application note:

http://openpicus.googlecode.com/files/FLYPORT%20Programmer%27s%20Guide%202.2%20release%201.0.pdf

THIS PAGE BLANK

Chapter 3 – Programming and downloading a project

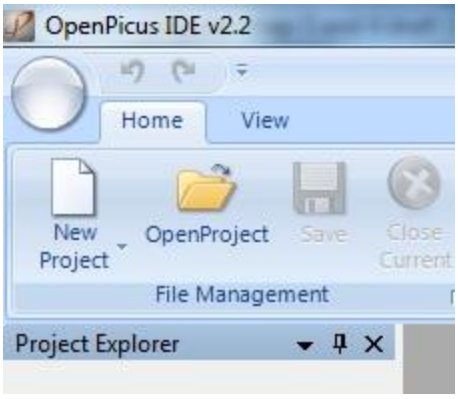
This chapter is split into two parts. The first part walks the user through loading the code onto the Flyport mission computer. The second part is about interacting with the code on the Flyport.

Part 1: Loading the Code

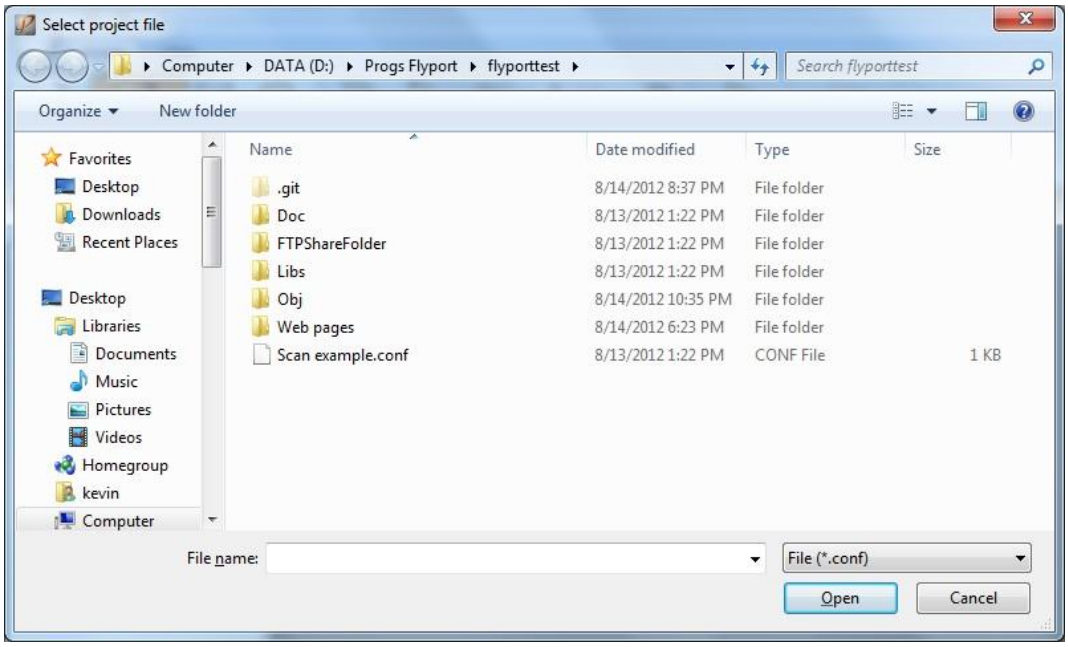
Starting the IDE

Choose "OpenPicus Flyport IDE 2.2" from the Windows start menu below.

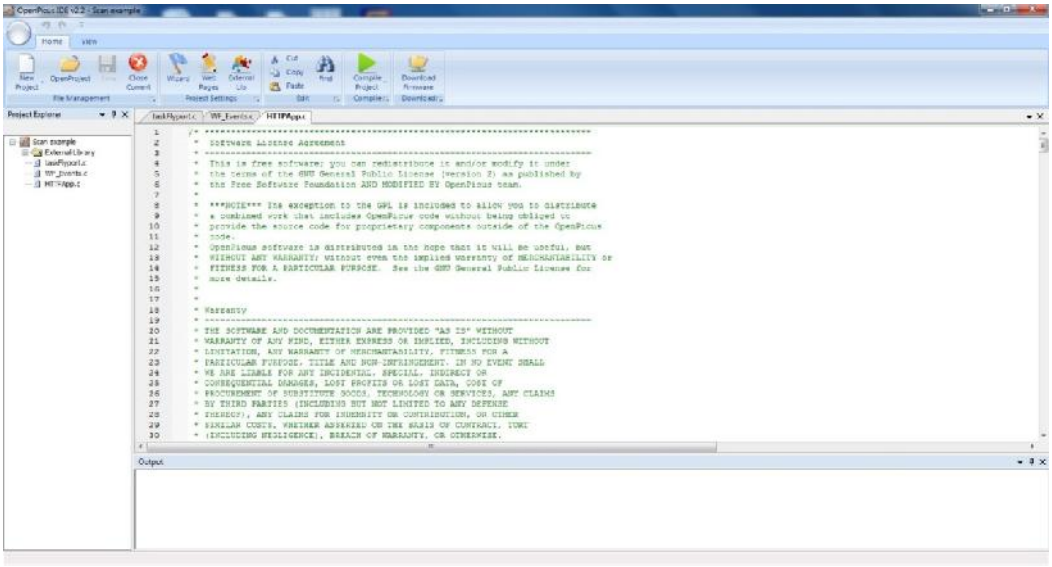
Choose OpenProject above the Project Explorer, window in the upper left hand corner of the screen.



The select project file dialog box opens. Choose the ".conf" file. It is the one that contains project information. In our case it is "Scan example.conf" as shown below.



The IDE will open as shown below. It will open all relevant tabs in the main window pane. The project explorer is on the left-hand side of the IDE. The output dialog box is on the bottom. The output dialog box is where you will receive messages from the compiler. The serial monitor is not yet open. We will open the serial monitor later when we download the program to the Flyport.



TaskFlyport.c is the file where the main task taskFlyport() is located. This is where the program starts. The code that pertains to the user, starts in taskFlyport() and where you will be spending most of your time. We do not recommend changing any other portions of the code as it could cause the Flyport to stop working.

Programming a project

The programming language used to program the Flyport is C. C lacks many of the object oriented and variable passing qualities found in C++. Still, it is a powerful language.

Choose the taskFlyport.c tab to show the taskFlyport() function. The main taskFlyport() function reveals several things. Notice the MAIN LOOP START. It is a continuous loop that keeps running until the Flyport is turned off. Within that loop there are five separate processes (lines 143 to 217). Only one process will run each time through the loop. Which process will run is based on the many true or false variables in each if statement beginning each process. It is set up so that only one process will run each time the loop cycles. The exception to that is the delay process.

- **The delay process** – This is a timer function that gets called every time the while(1) loop repeats. It is used primarily to delay the scan process.
- **The scan process** – This process calls the WFScan() function. Its recent scan results can be accessed with the WFScanList() function. It also sets ScanCompleted to TRUE when it is finished scanning.
- **The data manipulation process** – This process calls the DataManip() function.
- **DataManip** – SURVEY Mission: This mission is used only to find available field network stations. The mission collects scan data, essentially WiFi infrastructure information, and place this data in the array of NetNumber_tWFNetwork structures.
- **DataManip** – COLLECTION Mission: This mission is used to collect data from field network stations, and bring them back to the home network station. The collection network is found by comparing the current scan results to a list. This list contains the field network stations added from the Web server page (see Collection Setup). When the current scan result reveals a network on the list, it collects the field network station's data. DataManip() calls the readOrWriteToFTP() to do the collection.
- **The dump memory process** – This process calls to readOrWriteToFTP() function. It is used in a SURVEY mission, to dispense the scans stored in memory to the home network station. When this function gets called each field network station it found and stored in memory gets uploaded one at a time to the specified home network station file. The home network station is automatically detected and the upload starts by itself.
- **The write collected data process** – This process calls to readOrWriteToFTP(). It is used in a COLLECTION mission to write data to the home network station once it returns. Here again, the home network station is automatically detected. If no data was collected from the field network stations, it will write an error message to the file.

Definition of terms

The "home base station" is where the main FTP server is located. It is where the quads return from a survey mission or a data collection mission.

The "field network station" is a remote WiFi-based weather/environmental station deployed in the field.

Compiling a project - Overview

During a project's compile, the Flyport IDE's compiler takes the .c source code files, and converts them to object files (.o). It then links all of these together, and creates a .hex file. The user should know that a .hex file is actually an ascii character file. It uses the characters “0” to “9” and “A” to “F” to represent machine code in a hexadecimal ascii character format. The hex file will then be used to make a machine object file that will be run on the Flyport itself.

There are many object files already compiled in the OpenPicus directory. This is where the OpenPicus was originally installed. Usually it is in your program files, for us it was the following directory:

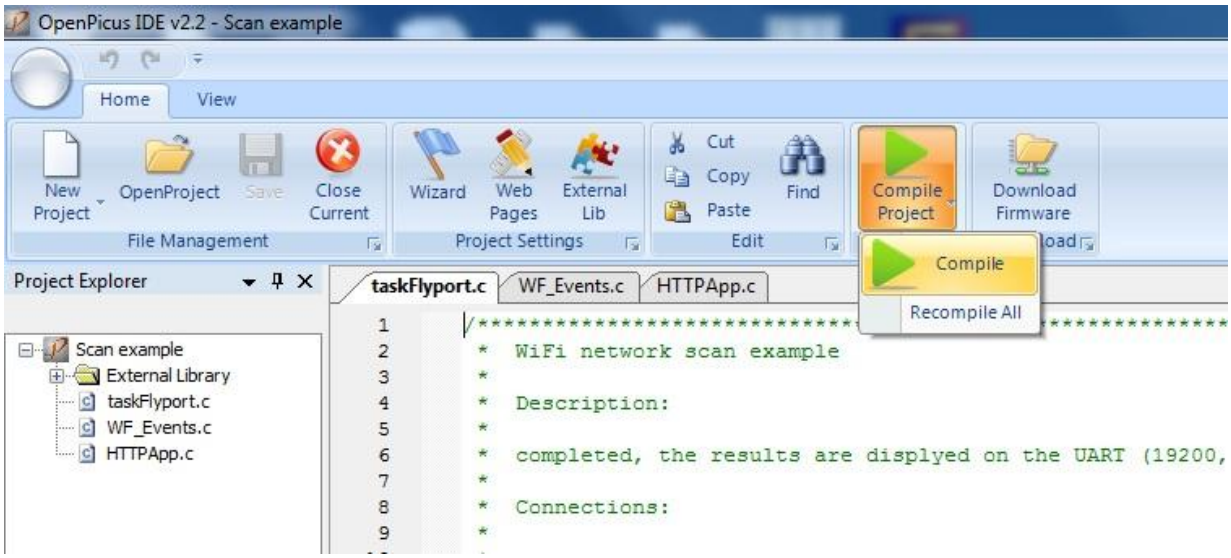
C:\Program Files (x86)\OpenPicus Flyport IDE 2.2

These files are already compiled, and don't have to be recompiled again each time. The user only needs to worry about the code in his project directory. We started our project in the following directory:

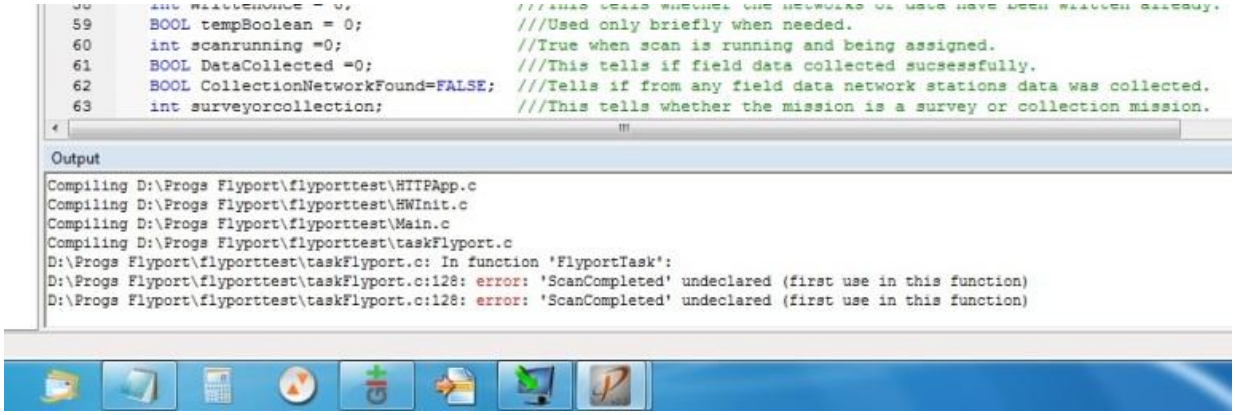
C:\Users\owner\Desktop\ProgsFlyport\Flypporttest

Compiling a project

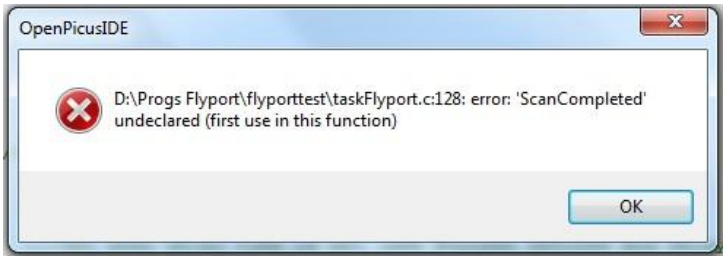
Choose Compile Project on the Home tab ribbon as shown below. Then choose Compile. Occasionally, the user will notice errors from the Program Files directory. This is an indication that you need to choose Recompile All, instead of just Compile Project.



Look in the output window, you will see it listing the files as it compiles. If there are errors, as in this case, you will see them listed too.



You will also see below the error dialog box. Click OK and fix your errors.



When you have successfully compiled, it will list the amount of memory used. You may have to scroll the output window to see all of the output. The top number is flash memory used, the bottom number is RAM. You have 256K of flash, and 16K of RAM. If the bottom number is greater than 100 it will give you an error, and you will have to change your memory allocation. In the file MissionParameters.h line 43, you can change your MAXNETWORKS to a smaller or larger number if necessary.

```
-------- OpenPicusIDE: Start Project Compilation --------

Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP_Utils.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\WF_Utils.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\Announce.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\ARP.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\Delay.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\DHCP.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\DHCPs.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\DNS.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\DNSs.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\DynDNS.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\ENCX24J600.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\FTP.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\Hashes.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\Helpers.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\HTTP2.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\ICMP.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\IP.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\MPFS2.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\NBNS.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\Random.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\Reboot.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\SMTP.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\SNTP.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\SPIFlash.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\StackTsk.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\TCP.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\Tick.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\UDP.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP
Stack\ZeroconfHelper.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP
Stack\ZeroconfMulticastDNS.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP
Stack\WiFi\WFConnectionAlgorithm.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP
Stack\WiFi\WFConnectionManager.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP
Stack\WiFi\WFConnectionProfile.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP
Stack\WiFi\WFDataTxRx.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP
Stack\WiFi\WFDriverCom.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP
Stack\WiFi\WFDriverRaw.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP
Stack\WiFi\WFEventHandler.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\WiFi\WFInit.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\WiFi\WFMac.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP
Stack\WiFi\WFMgmtMsg.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP
Stack\WiFi\WFParamMsg.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP
Stack\WiFi\WFPowerSave.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\WiFi\WFScan.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP
Stack\WiFi\WFTxPower.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP
Stack\WiFi\WF_Config.c
```

```
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\WiFi\WF_Eint.c
Compiling C:\Program Files (x86)\OpenPicus FlyPort IDE 2.2\Microchip\TCPIP Stack\WiFi\WF_Spi.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\HTTPApp.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\HWInit.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Main.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\taskFlyport.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\WF_Events.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\MPFSImg2.s
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\ExternalLib\ConnectionHelper.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\ExternalLib\FTPClient.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\ExternalLib\FTPHelper.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\ExternalLib\StringFunctions.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\Flyport libs\ARPlib.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\Flyport libs\FTPlib.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\Flyport libs\HWlib.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\Flyport libs\INTlib.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\Flyport libs\ISRs.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\Flyport libs\NETlib.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\Flyport libs\Regs.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\Flyport libs\SMTPlib.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\Flyport libs\TCPlib.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\Flyport libs\UDPlib.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\FreeRTOS\heap_2.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\FreeRTOS\list.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\FreeRTOS\port.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\FreeRTOS\queue.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\FreeRTOS\tasks.c
Compiling C:\Users\owner\Desktop\Progs Flyport\Flypporttest\Libs\FreeRTOS\portasm_PIC24.S

*                    Total program memory used (bytes):        0x2e3dd  (189405) 72%

*                       Total data memory used (bytes):         0x3ab8  (15032) 91%

------------------------------ End Project Compilation ------------------------------------
---------- Started at 14-Sep-12 04:14:29
---------- Completed at 14-Sep-12 04:18:01
---------- Duration (hh.mm.ss) 00.03.32
-------------------------------------------------------------------------------------------
```

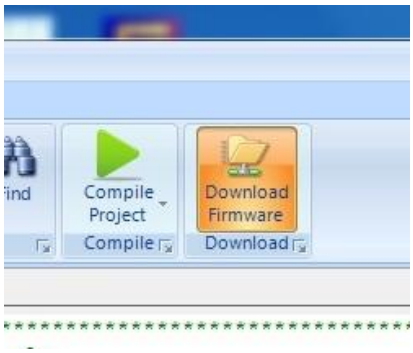
You are now ready to download your project.

Downloading a project - Overview

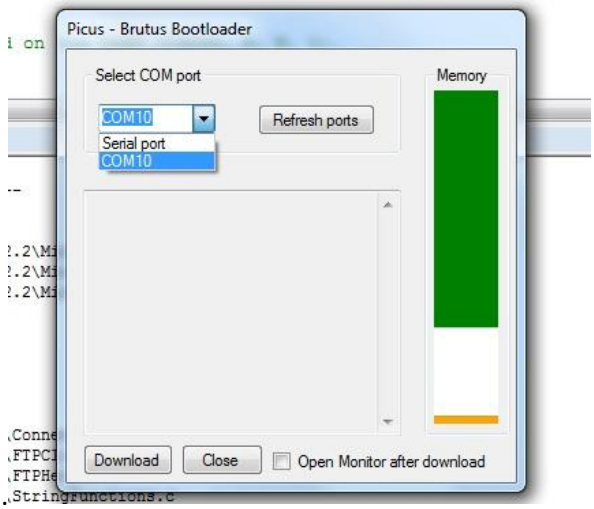
By downloading a project, the openPicus IDE takes the .hex file, parses it, and turns it into machine code that the Flyport can understand. It then downloads it to the Flyports flash memory (256K). As said before, the user should know that a .hex file is actually an ascii character file. Parsing takes these characters, and changes them to binary numbers that become the machine code. The IDE then downloads it. It takes a few minutes for this to happen then the Flyport automatically executes the program.

Downloadng a project

Plug the Flyport into the development board, to an open USB port on your computer or tablet. Choose the Download Firmware button on the top ribbon of the IDE as shown below.

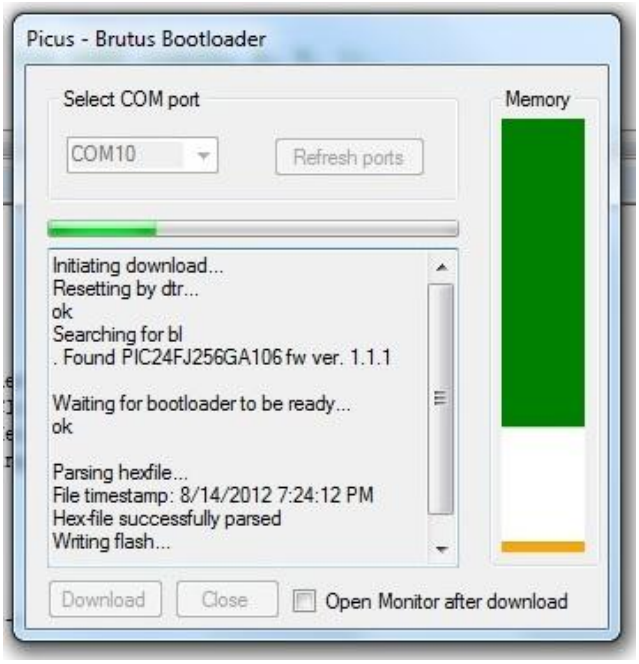


This opens the Picus - Brutus Bootloader screen as shown below.

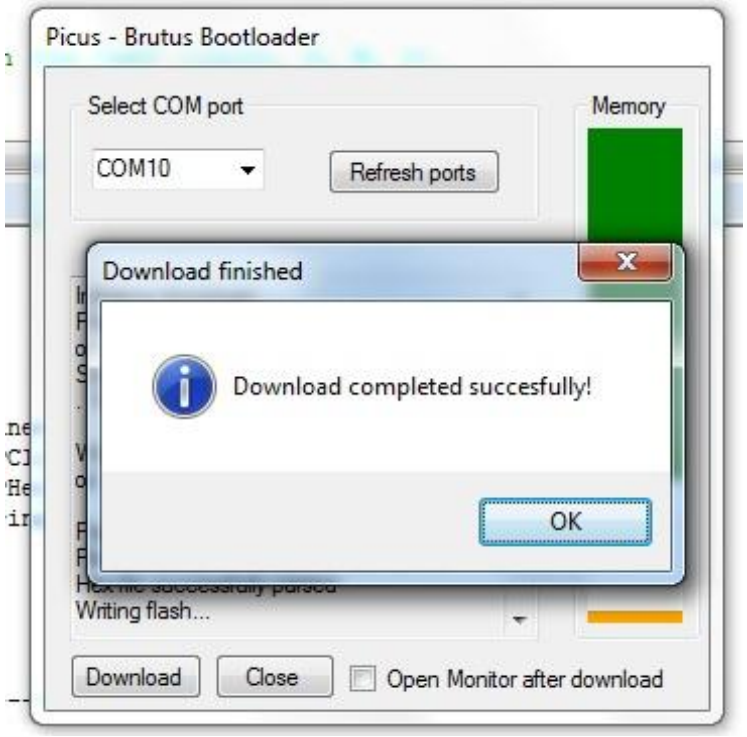


Click refresh ports, and then choose an available COM port. Each board is assigned a different COM port, and this may involve downloading the correct Flyport driver. Click the Download button.

The user will see the progress bar move to all green as it is downloading as shown below.



When it is finished, it will say “Download completed successfully!”

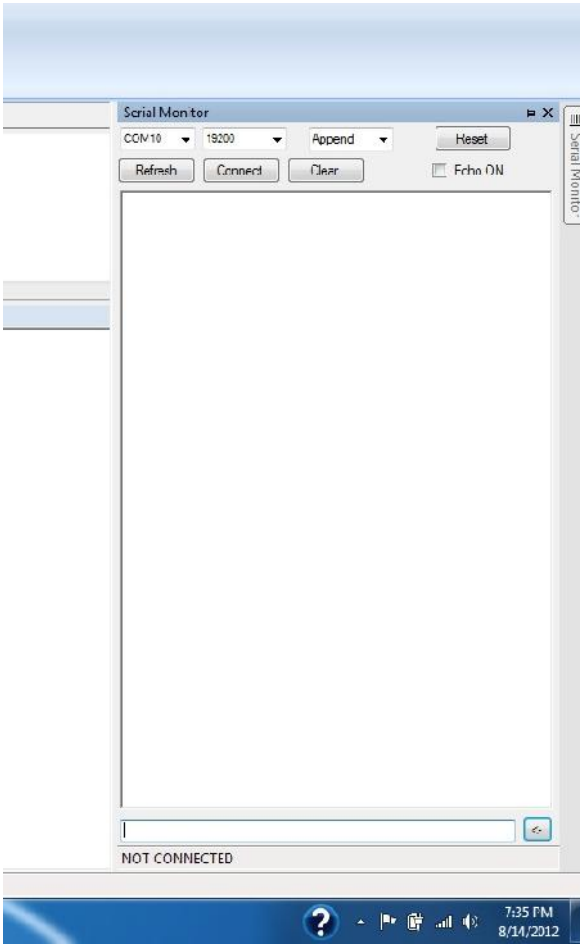


```
Initiating download...
Resetting by dtr...
ok
Searching for bl
Found PIC24FJ256GA106 fw ver. 1.1.1

Waiting for bootloader to be ready...
ok

Writing flash...
ok
Download completed! Download finished
```

You are now ready to open the Serial Monitor.

Choose the view tab in the upper left corner of the IDE. This allows the user to select the Serial Monitor button.

The serial monitor will open as shown below. If it closes automatically, place your mouse cursor over the Serial Monitor Tab on the screen's right side. Notice at the top the three dialog boxes and the reset button. Below that are the Refresh, Connect and Clear buttons, as well as the Echo ON check box. On the bottom of the screen, there is a box for entering input. Below that tells whether or not the Monitor is connected to the Flyport.



Click the Refresh button to refresh the COM ports available. Choose the one for your Flyport, and then choose 19200 and Append in the other dialog boxes. You can then choose Connect. Don't be worried about the Overrun message (see below), this simply means that the Flyport was producing output before it was connected. Click the Reset button and you should see the initialize message from your Flyport. The Flyport will then begin executing code that was coded in taskFlyport() function. You are now ready to prepare the mission project.

Part 2: Interacting with the Flyport

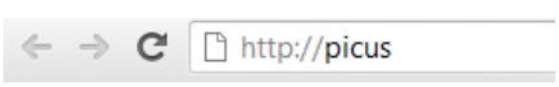
Preparing a project - Introduction

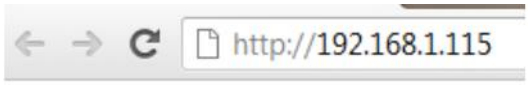
A web page is used as the user interface for programming the onboard mission computer. All interaction with the mission computer is done through this web page. The page will allow the user to set up a home network for the mission computer to connect to. It will also let a user set any field network stations where data will be collected from. Finally the user will control when to start the mission.

Initial Use

The mission computer is programmed to set up an AdHoc network as its default behavior. For this reason the first time the mission computer is turned on the user must connect to the mission computer's AdHoc network to interact with it. The default AdHoc network SSID name is "FlyportNet" with security set to open. The user can connect to this network just like any other WiFi network.

Once connected to "FlyportNet" the user can interact with the mission computer by using a web browser. The web page can be reached at http://picus or by IP http://192.168.1.115.



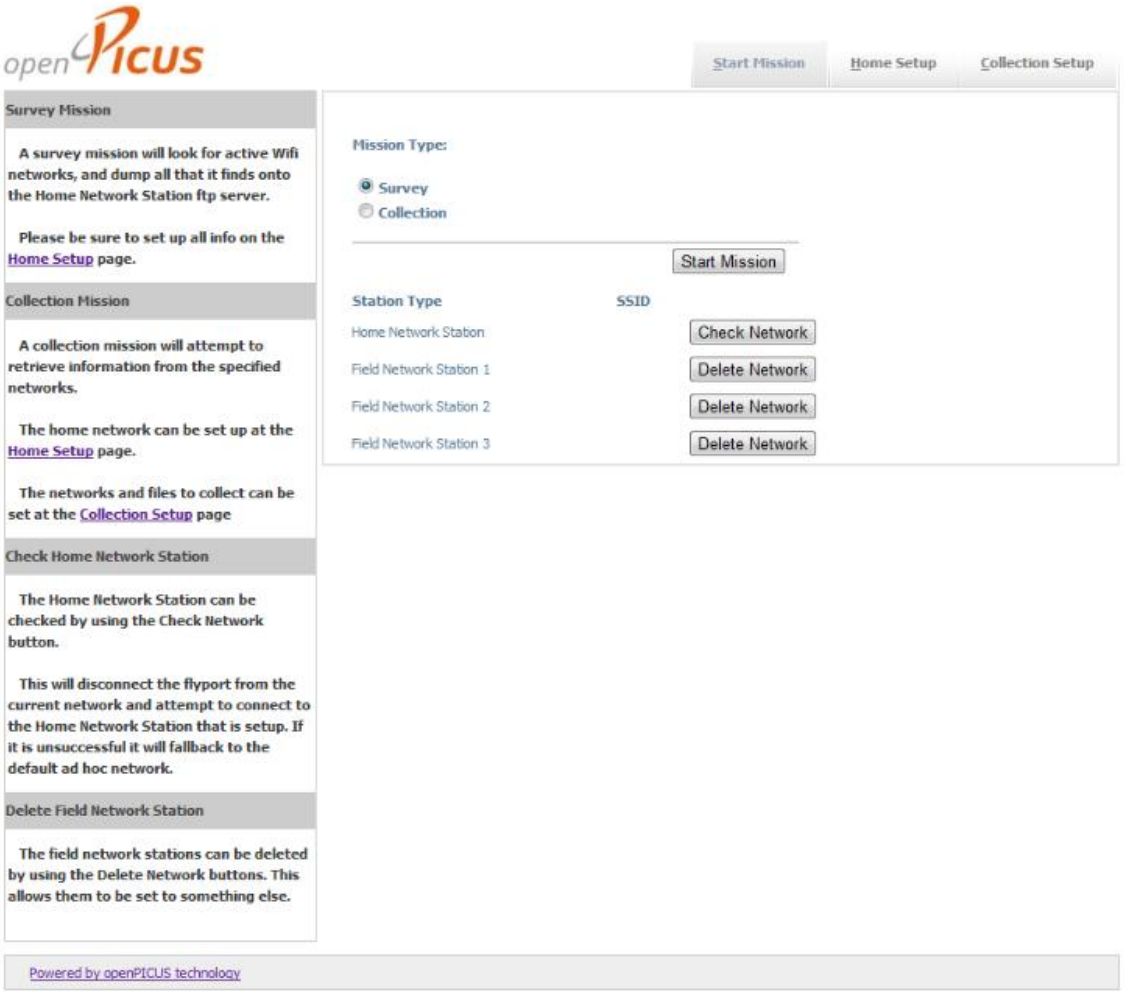


Overview

After using the webpage to setup the home network station and after the mission computer successfully connects to the home network station, it will continue to use that network as its default and consequently the initial set up of connecting to the "FlyportNet" AdHoc network will not be needed. However if the mission computer fails for any reason to connect to the home network station it will default back to the "FlyportNet" AdHoc network and the initial use case will be needed again.

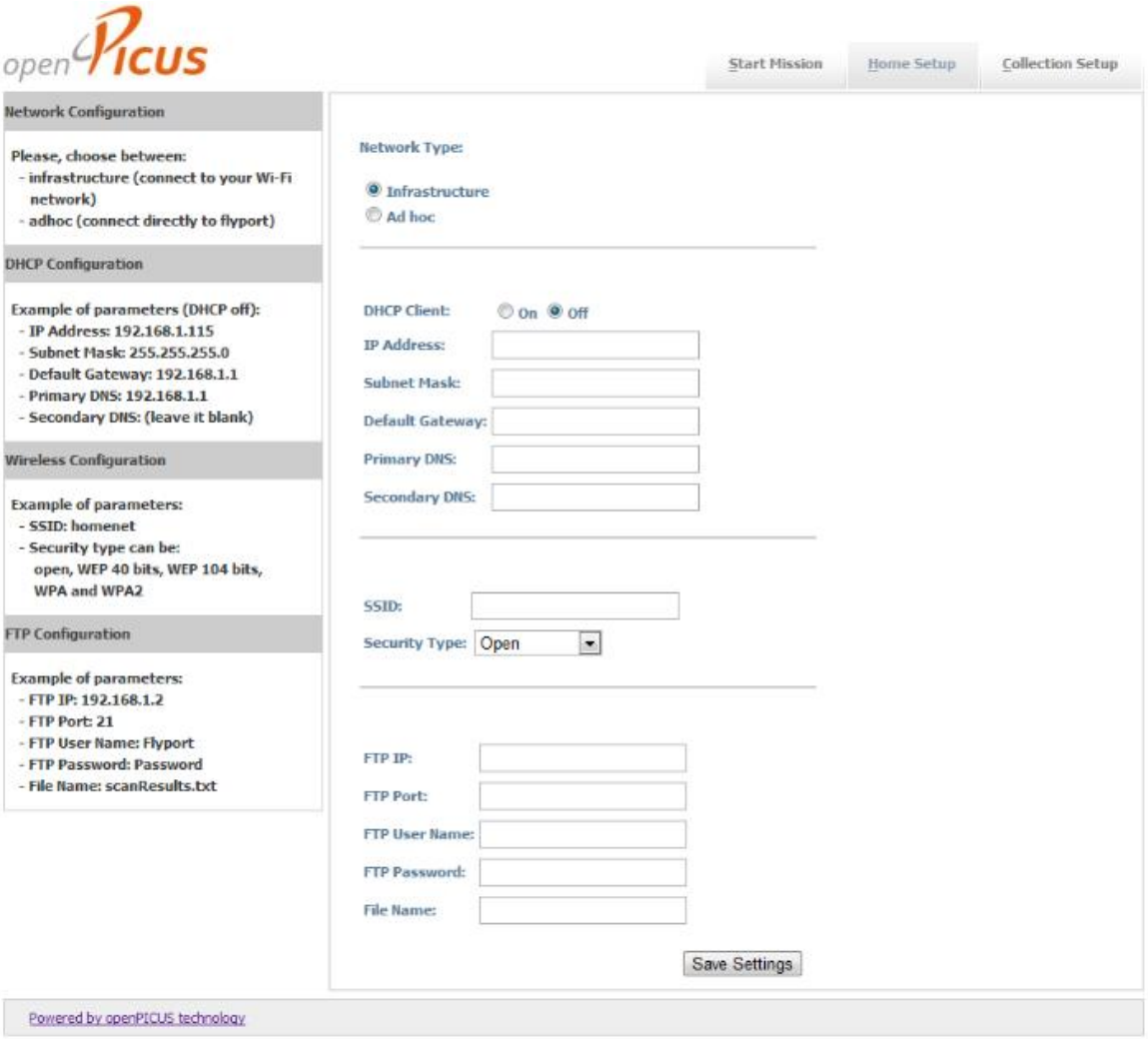
Normal Use

If the mission computer connected to its home network station during its last mission it will attempt to connect to that network again when it is powered on. The user will then have to connect to the same home network station and navigate to the mission computer's web page via a web browser. The web page can be reached at http://picus or by IP. The IP will be whatever the user set the home network IP to or be assigned by DHCP which may vary from connection to connection. Below is the webpage.



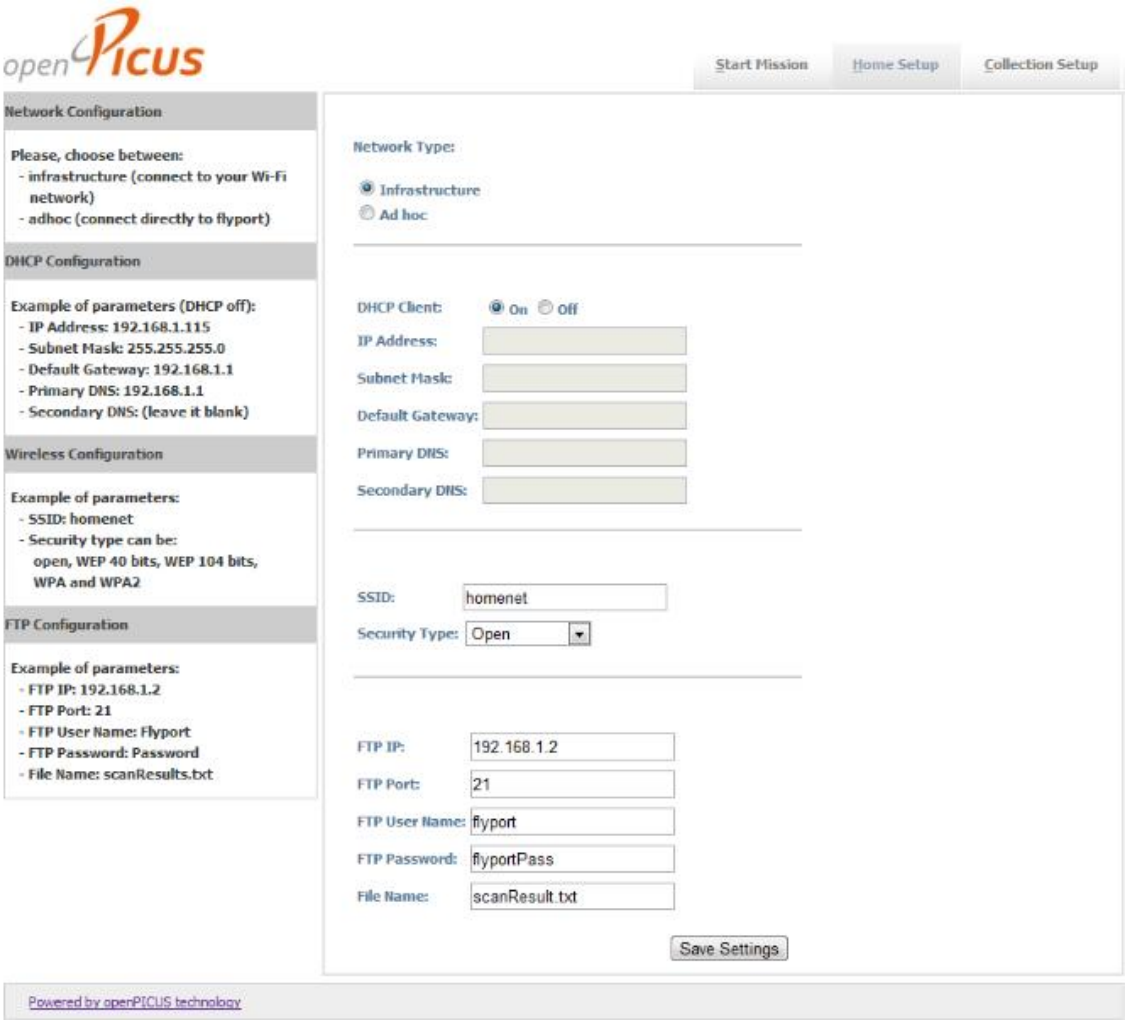
Home Setup

This page is where a user sets all the information for the home network station FTP server. The home network station is the network that the mission computer will connect to after performing its mission, and default to when it powers on. The FTP server information is for the home network station's FTP server where the mission computer will upload any data it collects. The file name is the name of the file that will contain the results of a survey mission.



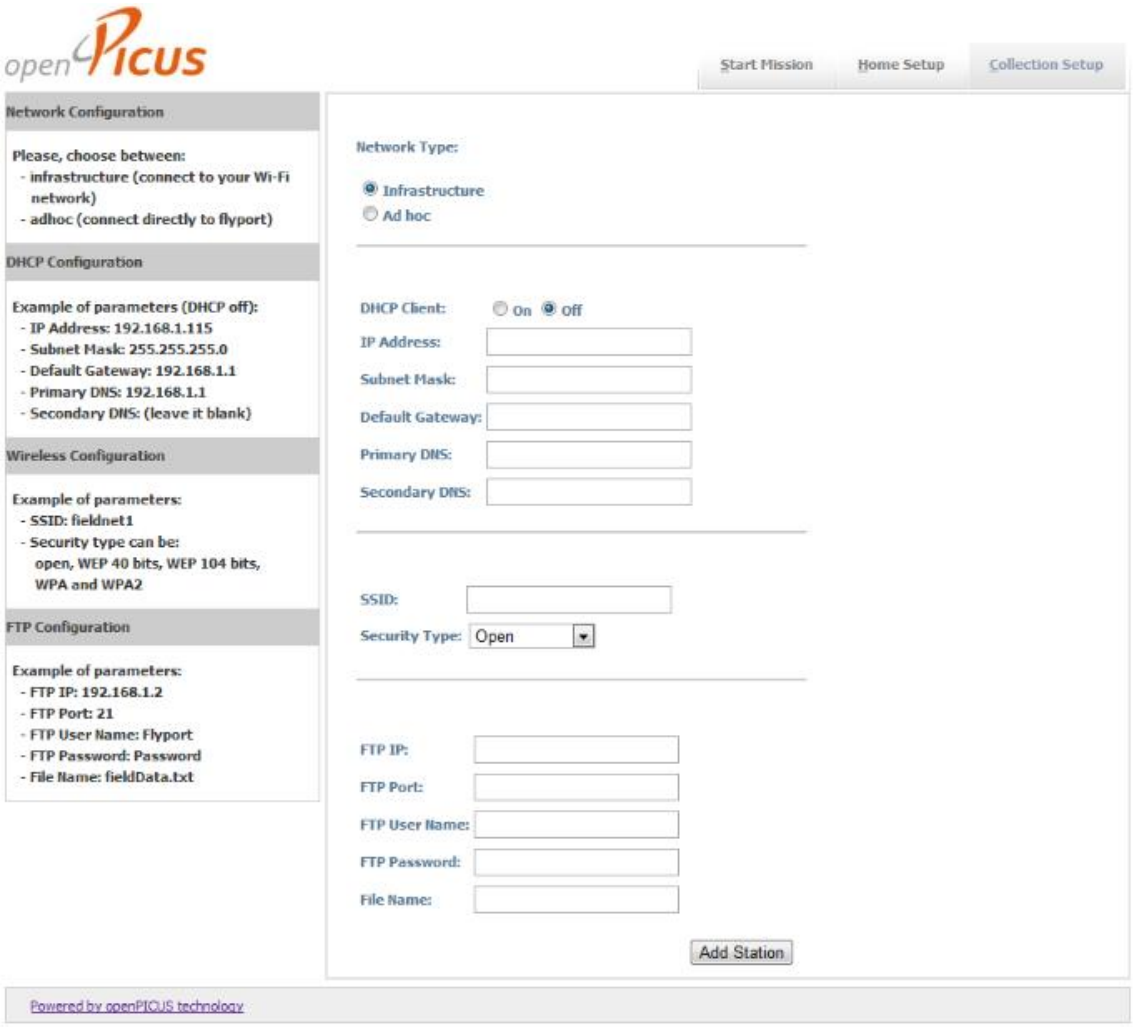
Below contains example information for a home network station. The user must click Save Settings for this information to be saved to the mission computer. All fields are required, and if any are not filled in an error message will be displayed.

In the example, the mission computer will connect to an infrastructure network using DHCP with and SSID of “homenet” using open security. It will use an FTP server with an IP of 192.168.1.2, port of 21, user name of “Flyport”, and password of “FlyportPass”. If sent on a survey mission the results will be uploaded on the FTP server to a file with the name of “scanResult.txt”.

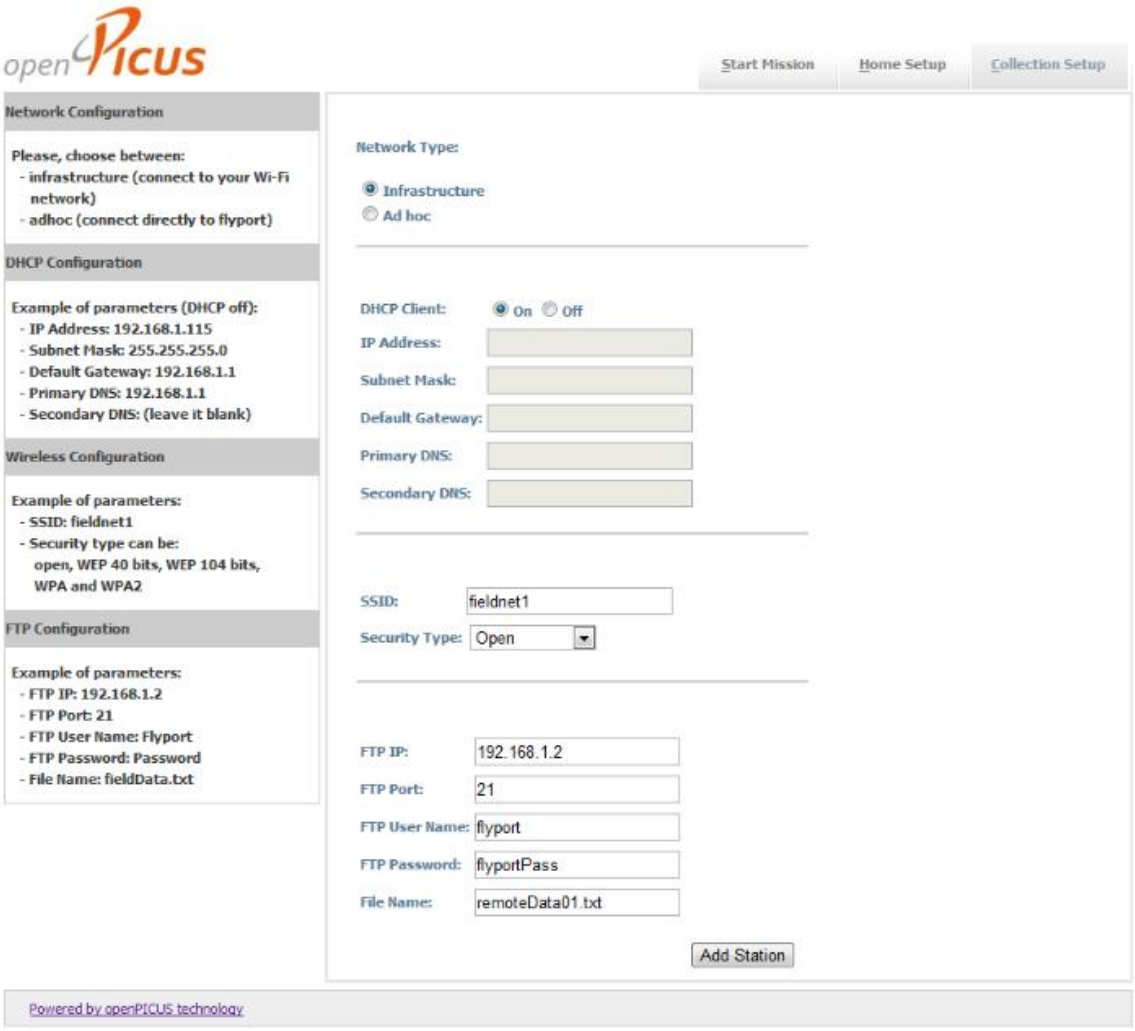


Collection setup

This page is used to add field network stations that will be used during a collection mission. It requires all the same information as a home network station. The mission computer will attempt to connect to all networks added this way and download the file specified by the File Name field.



Below contains example information for a field network station. The user must click Add Station for this information to be saved to the mission computer. All fields are required, and if any are not filled in an error message will be displayed.



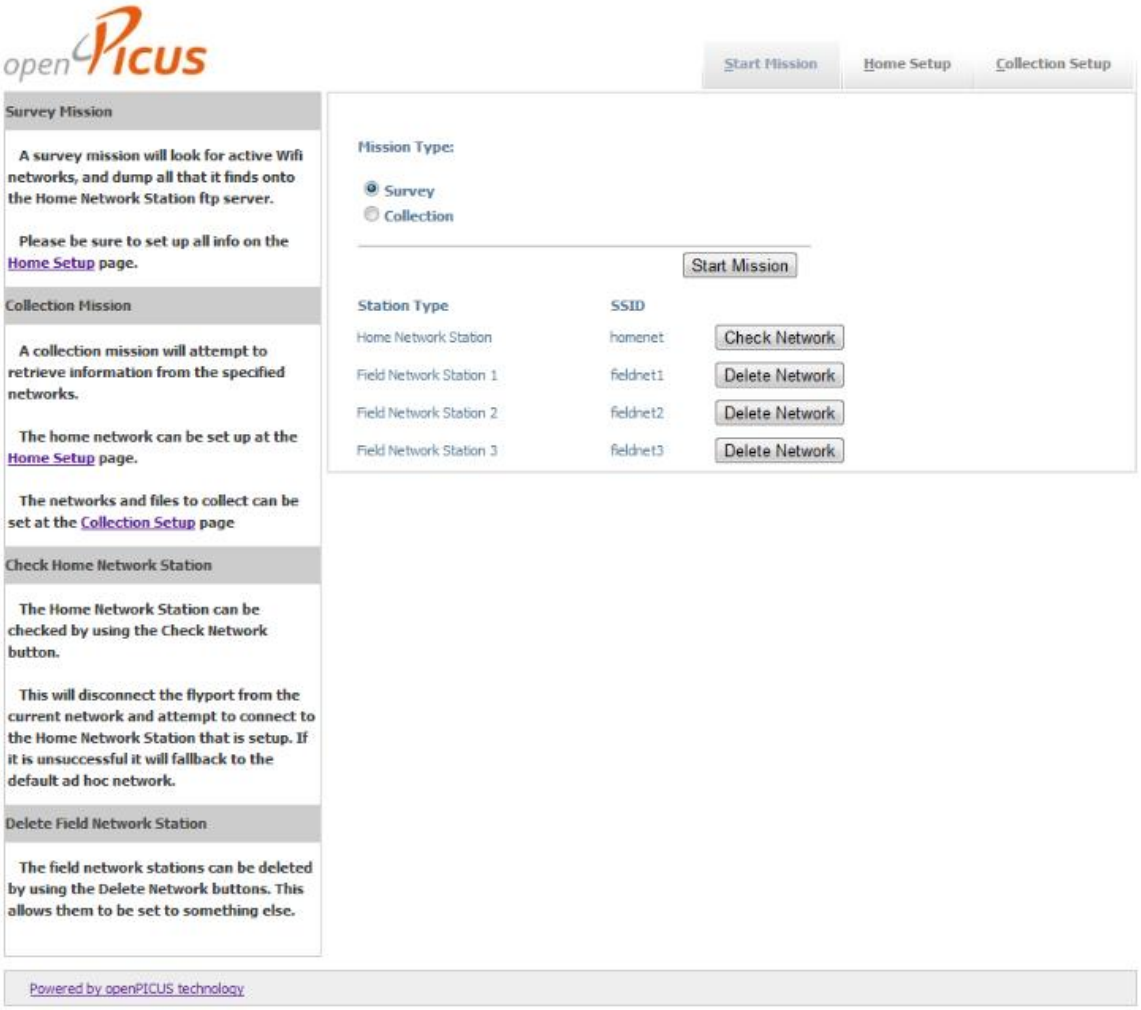
In the example, the mission computer will connect to a field infrastructure network using DHCP with and SSID of “fieldnet1” using open security. It will use an FTP server with an IP of 192.168.1.2, port of 21, user name of “Flyport”, and password of “FlyportPass”. If sent on a collection mission it will download “remoteData01.txt” from the “fieldnet1” field network station and upload that file on the home network station FTP server with the same file name.

Currently due to program constants there is a strict limit of three field network stations. Attempting to add any more than three will result in an error. If the user wishes to change one field network station they can delete a field network station from the Start Mission page, which will make room for an additional field network station.

Note: Currently any data collected from the given file is limited to 200 characters. It could be increased if the mission computer is equipped with external storage such as an SD card. This would require additional programming.

Starting the Mission

Below shows an example Start Mission page with the home network station SSID set to “homenet” and three field network stations with their SSID's set to “fieldnet1, fieldnet2, and fieldnet3”.



The user can check that the settings for the home network station are correct by using the Check Network button. This will disconnect the mission computer from the current network and attempt to connect to the home network station. If successful the mission computer will then be connected to the home network station, if not it will fall back to the default AdHoc network (see Initial Use).

Any field network station can be deleted by using the Delete Network button next to the field network station to be deleted. This is useful to make room for other networks, or to correct an error.

When the user is ready to start a mission they may select either survey or collection mission type and press the Start Mission button. The survey mission will not start unless the home network station is setup, and the collection mission will not start unless the home network station and at least one field network station are setup.

Chapter 4 – Flying a project on the onboard mission computer

Flying a project

After the Start Mission button is pressed the mission computer will begin performing its mission.

If survey mission was selected it will scan for WiFi networks and save the first 40 that it finds. After a set period of time it will start looking for its home network station SSID, and when it finds it will connect to it. If it successfully connects to the home network station, it will upload via FTP all network information that it found to the file name setup in Home Setup.

If collection mission was selected it will look for any of the specified field network station SSIDs. Once it finds one it will attempt to connect to that network and FTP server. If successful the mission computer will download the file that was set with file name on the Collection Setup page. It will do this for all field network station SSIDs that it finds. Again after a set period of time the mission computer will start to look for the home network station. When it finds the home network station it will attempt to connect to it and the home network station FTP server. If the connection is successful it will upload all files that it downloaded to the home network station FTP server.

Note: In the program it should be possible to change it from waiting a set period of time to waiting until at least one field network station is found, before looking for the home network station. This is currently not feasible because there is no way to store all field network stations in RAM.

After either successful mission the mission computer will be able to be connected to again. Then the user may change settings and start another mission.

THIS PAGE BLANK

Chapter 5 – The FileZilla FTP server

Setting up and testing the FileZilla FTP server

Setting up the FTP server is very similar to setting up the client, with the main difference being that the roles of active and passive mode are reversed.

A common mistake, especially by users with NAT routers, is in testing the server. If you are within your local network, you can only test using the local IP address of the server.

Passive mode

The server configuration is very similar to the client configuration for active mode. In passive mode, the server opens a socket and waits for the client to connect to it.

If you have a NAT router, you need to tell FileZilla Server your external IP address or passive mode connections will not work with clients outside your local network:

1. If you have a fixed external IP address, you can enter it in the configuration dialog of FileZilla Server, or
2. If you have a dynamic IP address, you can let FileZilla Server obtain your external IP address from a special website automatically. Except your version of FileZilla Server, no information will be submitted to that website.

If in doubt, use the second option.

Setting up FileZilla server with Windows Firewall

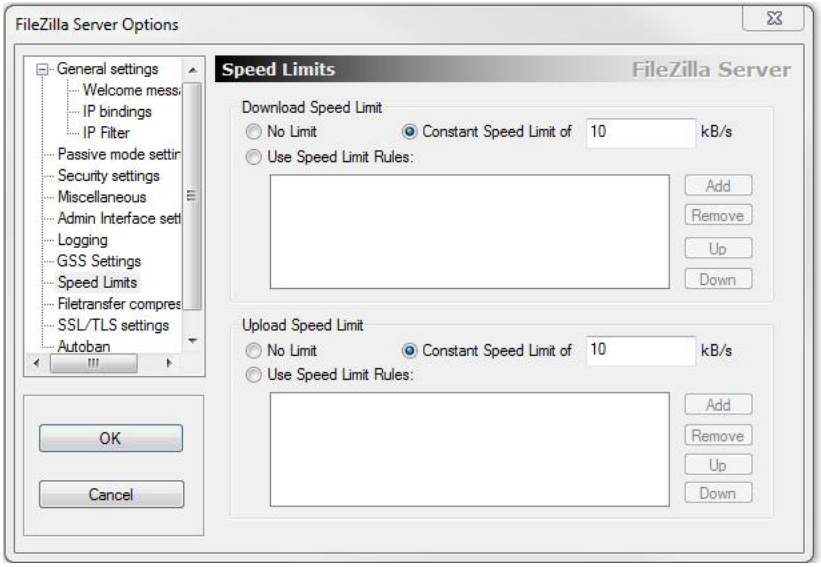
If you are having problems with setting up FileZilla Server to run behind Windows Firewall (specifically, it fails on "List" and the client receives a "Failed to receive directory listing" error), you must add the FileZilla Server application to Windows Firewall's Exceptions list. To do this, follow these steps:

1. Open Windows Firewall under Control Panel.
2. If using Windows 7, click “Allow a program or feature through Windows Firewall”.
3. Click “Allow another program...”
4. Do NOT select “FileZilla Server Interface” from the list, instead click on “Browse...”
5. Locate the directory you installed FileZilla Server to (normally "C:\Program
1. Files\FileZilla Server\")
6. Double click or select “FileZilla server.exe” and press open (Once again, NOT “FileZilla
2. Server Interface.exe”).
7. Select “FileZilla server.exe” from the list and click “OK”.
8. Verify that “FileZilla server.exe” is added to the exceptions list and that it has a check mark in the box next to it.
9. Press “OK” to close the window.

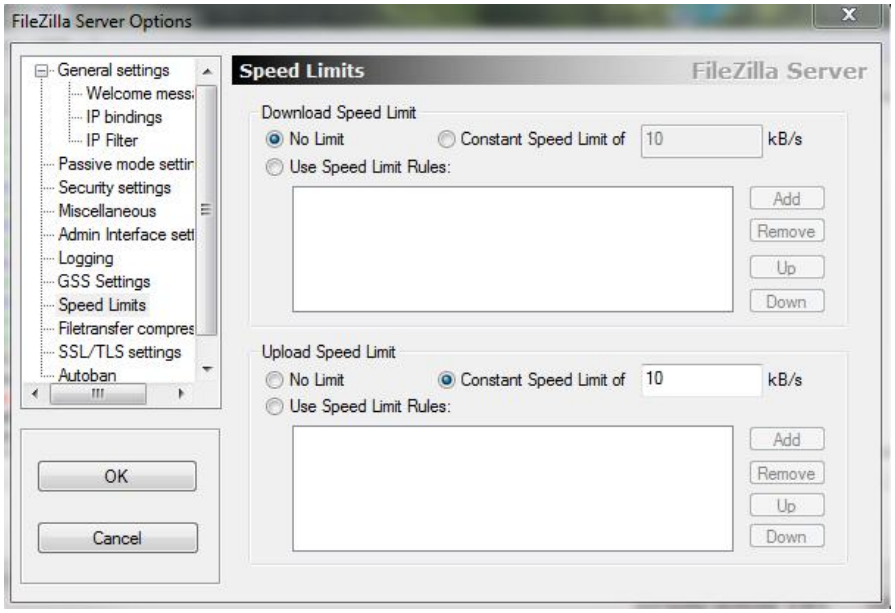
Passive mode should work now. If you are still having problems connecting (from another computer or outside the network), check your router settings or try to add the port number in the Windows Firewall settings.

FileZilla FTP Server

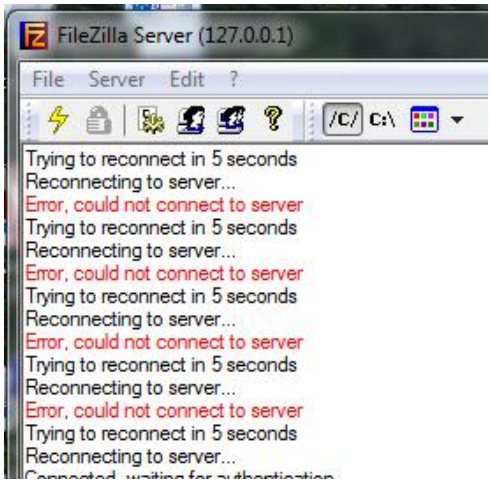
Since the Flyport speed limit is about 1 to 2Mbps, you can change the FileZilla Server speed limits from No Limit in both cases, to 10kps in both cases:



But changing it back to No Limit (the default) worked fine too:



NOTE: Using the Task Manager Services, it is possible to stop the FileZilla Server. However, when this is done, the FileZilla Server interface goes mad, see below:



For this reason it is better to keep the service running, you can turn the FileZilla Server on and off using the lightning bolt on the FileZilla user interface program:

ISSUE: When the FileZilla Server is simply left online, after a while, it seems that outside forces repeatedly attempt to log in on it:

Logged on
Creating listen socket on port 21...
Server online
(000004)5/8/2012 18:35:04 PM - (not logged in) (192.168.0.100)> Connected, sending welcome message...
(000004)5/8/2012 18:35:04 PM - (not logged in) (192.168.0.100)> 220-FileZilla Server version 0.9.41 beta
(000004)5/8/2012 18:35:04 PM - (not logged in) (192.168.0.100)> 220-written by Tim Kosse (Tim.Kosse@gmx.de)
(000004)5/8/2012 18:35:04 PM - (not logged in) (192.168.0.100)> 220 Please visit http://sourceforge.net/projects/filezilla/
(000004)5/8/2012 18:35:04 PM - (not logged in) (192.168.0.100)> disconnected.
(000005)5/8/2012 18:35:16 PM - (not logged in) (192.168.0.100)> Connected, sending welcome message...
(000005)5/8/2012 18:35:16 PM - (not logged in) (192.168.0.100)> 220-FileZilla Server version 0.9.41 beta
(000005)5/8/2012 18:35:16 PM - (not logged in) (192.168.0.100)> 220-written by Tim Kosse (Tim.Kosse@gmx.de)
(000005)5/8/2012 18:35:16 PM - (not logged in) (192.168.0.100)> 220 Please visit http://sourceforge.net/projects/filezilla/
(000005)5/8/2012 18:35:16 PM - (not logged in) (192.168.0.100)> disconnected.
(000006)5/8/2012 18:35:28 PM - (not logged in) (192.168.0.100)> Connected, sending welcome message...
(000006)5/8/2012 18:35:28 PM - (not logged in) (192.168.0.100)> 220-FileZilla Server version 0.9.41 beta
(000006)5/8/2012 18:35:28 PM - (not logged in) (192.168.0.100)> 220-written by Tim Kosse (Tim.Kosse@gmx.de)
(000006)5/8/2012 18:35:28 PM - (not logged in) (192.168.0.100)> 220 Please visit http://sourceforge.net/projects/filezilla/
(000006)5/8/2012 18:35:28 PM - (not logged in) (192.168.0.100)> disconnected.
(000007)5/8/2012 18:35:40 PM - (not logged in) (192.168.0.100)> Connected, sending welcome message...
(000007)5/8/2012 18:35:40 PM - (not logged in) (192.168.0.100)> 220-FileZilla Server version 0.9.41 beta
(000007)5/8/2012 18:35:40 PM - (not logged in) (192.168.0.100)> 220-written by Tim Kosse (Tim.Kosse@gmx.de)
(000007)5/8/2012 18:35:40 PM - (not logged in) (192.168.0.100)> 220 Please visit http://sourceforge.net/projects/filezilla/
(000007)5/8/2012 18:35:40 PM - (not logged in) (192.168.0.100)> disconnected.

It is possible that stopping the FileZilla Server service in the Task Manager, waiting a short while, then starting the service again appears to have stopped the above continual appearance of an outside force attempting to log in. Now things have been very quiet even though the server is online.

Appendix 1 – Current technical information

openPicus MICROCHIP IC24FJ256GA106

The PIC24FJ256GA106-I/MR is part of the PIC24F series family with 256kB Flash as a 16-bit Flash microcontroller. It can sustain standard temperature ranges from -40°C to +125°C and has 64 pins in a QFN square package.

Product Highlight

- **RAM Size:** 16 kB
- **Flash Size (Bytes):** 256 kB
- **No of I/O Lines:** 53
- **Speed:** 32 MHz
- **Supply Voltage:** 2 to 3.6 V

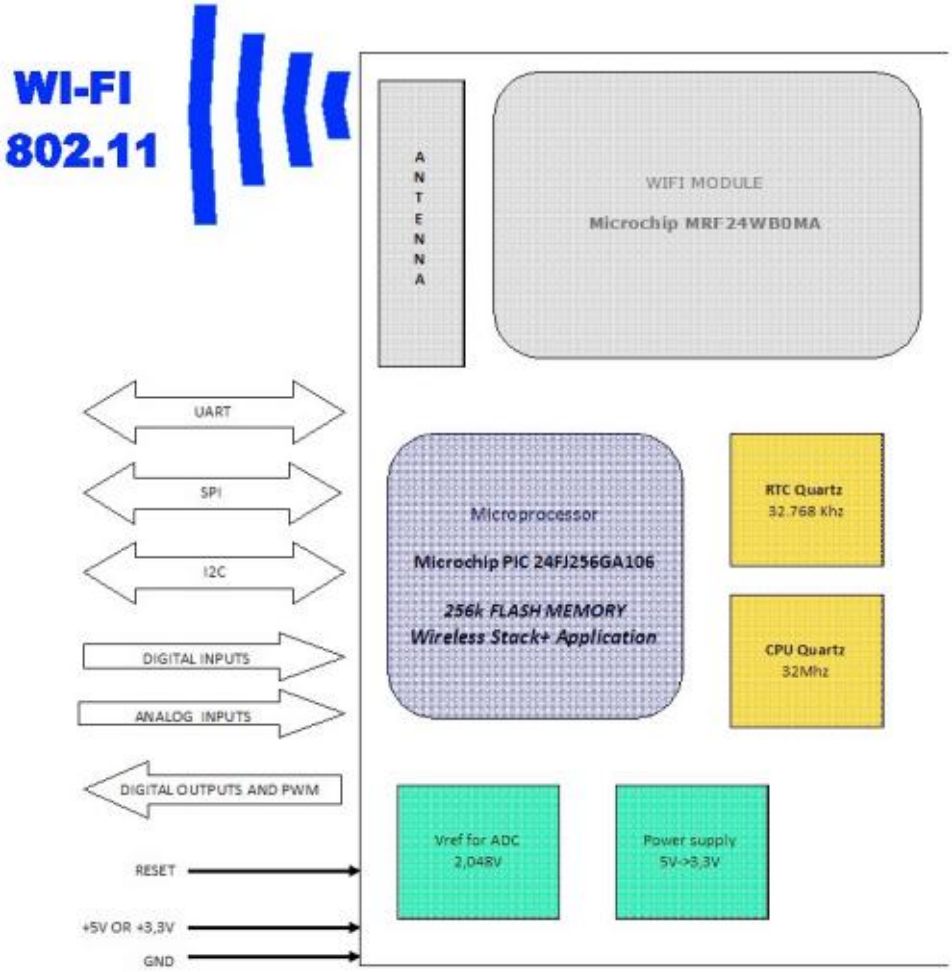
- **CPU**
  - Up to 16 MIPS performance
  - 16 x 16 hardware multiply, single cycle execution
  - 12-bit x 16-bit hardware divider
  - C compiler optimized instruction set
- **nanoWatt Power Managed Modes**
  - Run, idle and sleep modes
  - Multiple, switchable clock modes for optimum performance and power management
  - Run mode: 1 mA/MIPS, 2.0V typical
  - Sleep mode current down to 100nA typical
  - Standby current with 32kHz oscillator: 2.5uA, 2.0V typical
- **Flash Program Memory**
  - Self-reprogrammable under software control
  - 10,000 erase/write cycles
  - 20 year data retention
  - EEPROM emulation capable
- **System**
  - Internal oscillator support - 31kHz to 8MHz, up to 32MHz with 4X PLL
  - On-chip LDO voltage regulator
  - JTAG boundary scan and flash memory program support
  - Fail-safe clock monitor – allows safe shutdown if clock fails
  - Watchdog timer with separate RC oscillator
- **Analog Features**
  - 10-bit ADC, 16 channels, 500k samples per second
  - 3 analog comparators

The Flyport Wi-Fi is a miniature web server module featuring a fully integrated 802.11b/g/n WiFi interface and several interfaces to the "real world."

The module integrates a powerful 16-bit processor which runs custom applications and a WiFi certified transceiver which handles the connectivity. Two versions are available: one with PCB antenna and the other with a uFL connector for an external antenna.

The module provides the embedded world with a powerful "Internet engine" to a browser-based interface over Internet, in a small footprint, at low power and low cost. Real time data can be both displayed and/or updated from a standard web browser, even on smartphone or tablets, because the Flyport supports dynamic web pages

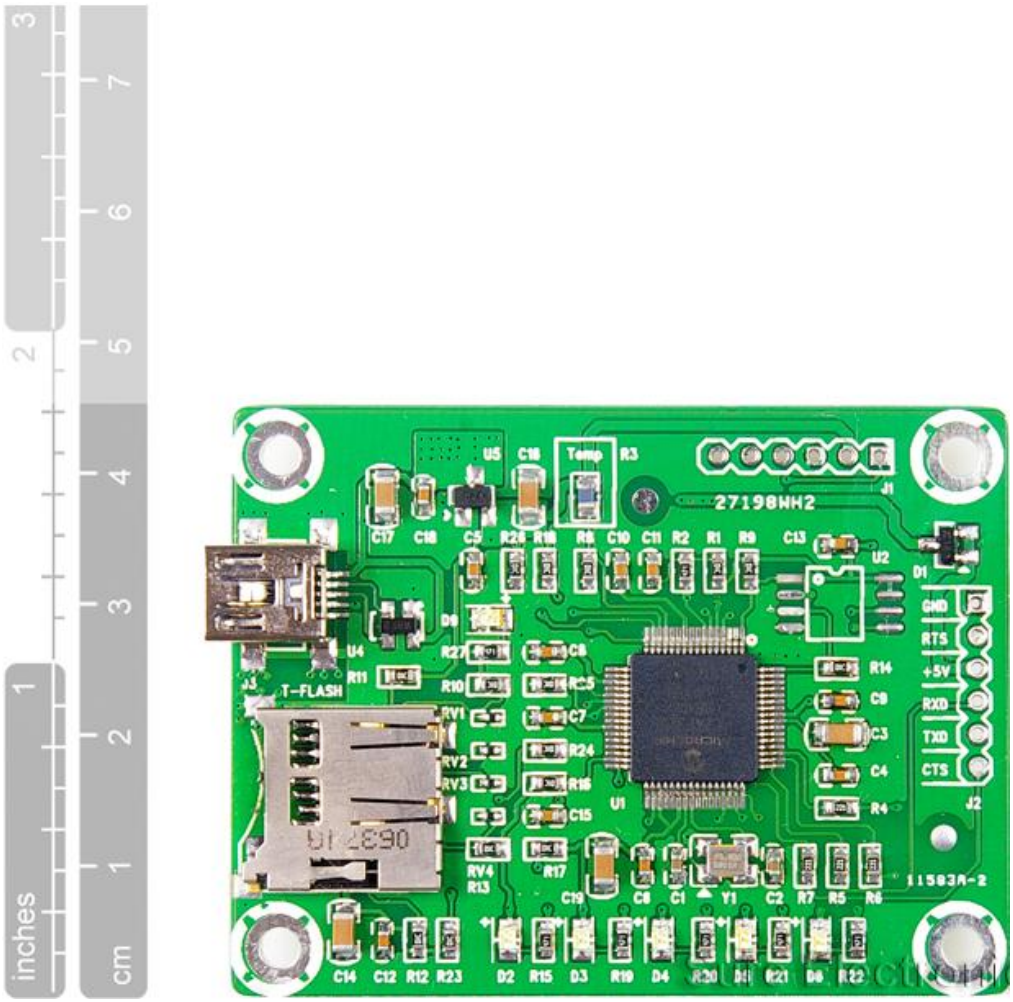
The Flyport is powered by the openPicus framework based on the FreeRTOS. The free penPicus IDE 2.2 allows creating applications, to import web pages and to compile and download code to the module.



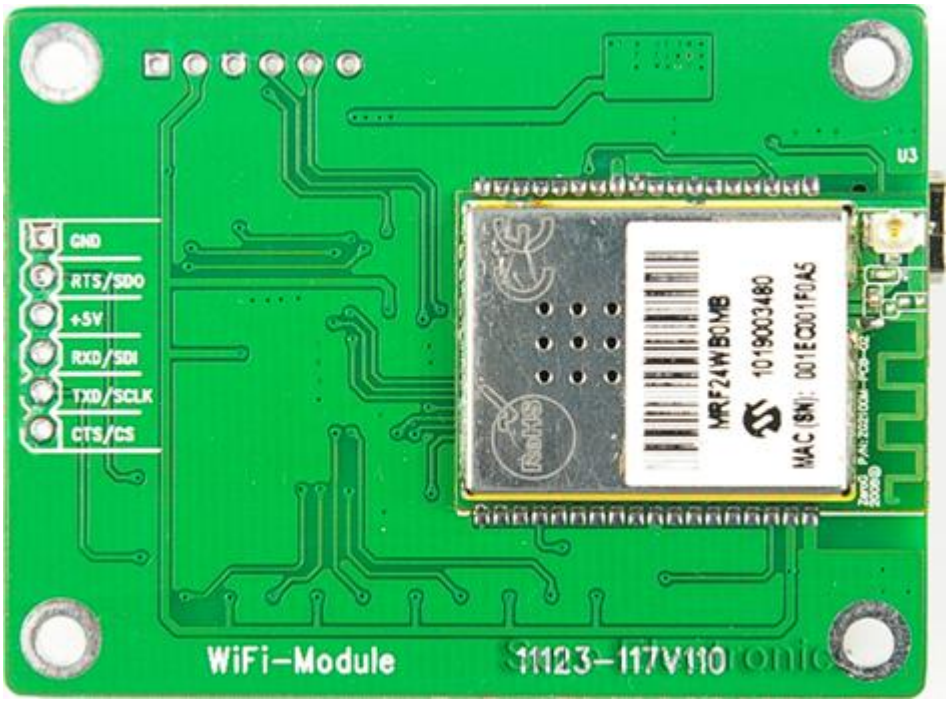
Appendix 2 – Evaluation of another mission computer

Sure Electronics

This is a demonstration board for evaluating WiFi connectivity using the PIC24FJ256GB106 and the MRF24WB0MB. MRF24WB0MB is an IEEE 802.11 WiFi radio transceiver module which has an uFL connector for an external antenna, matching circuitry, and supports Wi-Fi with the free TCP/IP protocol stack.



http://www.sureelectronics.net/goods.php?id=1185


GND
RTS/SDO
+5V
RXD/SDI
TXD/SCLK
CTS/CS
U3
MRF24WB0MB
WiFi-Module
11123-117V110

Features

- Size: 62.90 (L) × 45.70(W) ±0.2mm
- Powered by miniUSB cable
- Saving webpages in a microSD card
- Support dynamic page display
- WiFi wireless network, configurable via UART interface
- Web-based status monitoring
- Web-based LEDs control

Hardware

- PIC24FJ256GB106 IC
- MRF24WB0MB Wi-Fi radio transceiver module
- TF socket
- Five LED indicators
- miniUSB port
- A thermistor
- An external antenna
- UART interface – external serial port converter board required

Software

- Engineering source code
- Hex file
- Webpage

Package contents

- WiFi Web Server Module with TF Socket ×1
- MiniUSB cable
- Antenna
- 6-pin header

Results

We got 4 units in 2 separate shipments; Sure Electronics failed quality testing due to numerous hardware failures. It would have been nice to have an embedded microSD card reader for writing files.


Montgomery County
Community College