Manual for PWS Communications Module

ECE 4873 Senior Design

Aquanauts
Dr. Mick West
Dr. Robert Campbell

Shayna Seidel, EE, sseidel7, sseidel7@gatech.edu
Jim O'Donnell, CmpE, jodonnell3, jodonnell3@gatech.edu
Shelby Crisp, CmpE, scrisp3, scrisp3@gatech.edu
Jason Lee, EE, slee3117, slee3117@gatech.edu
Ruben Quiros, CmpE, rquiros6, rquiros6@gatech.edu
Timothy Pierce, EE, tpierce31, timrpierce31@gatech.edu

Table of Contents

1. Introduction and Intended Use …………………………………………………….. 4

2. Setup ……………………………………………………………………………. 5
2.1 Raspberry Pi, Cellular Kit, and SSD Instructions …………………………….. 5
2.2 PCB Instructions ………………………………………………………... 7

3. How to Use ………………………………………………………………... 9
3.1 Telit Commands ………………………………………………………… 9
3.2 Quectel Commands ……………………………………………………... 11
3.3 Benefits to Using the Quectel Chip …………………………………………... 13
3.4 Turning sensors on and off …………………………………………………… 13

4. Monitoring the Sixfab Cell Kit ……………………………………………..…… 16

5. References ……………………………………………………………………… 17

Appendix ……………………………………………………………………… 18

Nomenclature

CmpE: Computer Engineer

EE: Electrical Engineer

PCB: Printed Circuit Board

Pi: Raspberry Pi 3B

PWS: Prince William Sound

SSD: Solid State Drive

Manual for Communication Module

1. Introduction and Intended Use

This communication module is intended to be used for the Profiler located in Prince William Sound, which is monitored by Dr. Robert Campbell. This new module will replace the old Persistor processor and relay system with a Raspberry Pi 3B, Sixfab cell kit, and a MOSFET switching system on a PCB. The updated system will give the system more processing power for increased data collection, a stronger stable cellular connection to send more data using 3G/4G, and a subsystem that allows the research team to turn sensors on and off individually. Technological capabilities were increased, but with power consumption that is nearly similar to the original Persistor module.

2. Setup

The following parts are needed to run this prototyped system:

1) Raspberry Pi 3B with its external power supply (rated at 2.5 Amps and 5.25 V)
2) Minimum 16GB Micro SD Card
3) Samsung 870 EVO 250 GB SSD
4) SATA to USB cable
5) Sixfab cell kit with its external power supply (rated at 2.5 Amps and 5.25 V)
6) Printed Circuit Board (for details, see the Aquanauts' final report)
7) Micro USB to Micro USB cable

2.1 Raspberry Pi, Cellular Kit, and SSD Instructions

1. Flash both an SD Card and the SSD with your chosen version of Linux. More information can be found at the following link: Installing operating system images - Raspberry Pi Documentation Note: The SD Card is only required for initial setup, and is not required for permanent use of the Raspberry Pi.
2. For the Raspberry Pi 3B, a bit in the One Time Programmable memory must be set in order to enable booting from USB (booting from the SSD). This will require initially booting the Raspberry Pi 3B from an SD Card. More information can be found at the following link in the "**Raspberry Pi 2B v1.2, 3A+, 3B, Computer Module 3, 3+**" section: USB mass storage boot - Raspberry Pi Documentation
    a. For our chosen SSD, boot time took slightly longer than the default boot time of 2 seconds. This causes boot issues occasionally for the following reason:

USB enumeration is a means of enabling power to the downstream devices on a hub, then waiting for the device to pull the D+ and D- lines to indicate if it is either USB 1 or USB 2. This can take time: on some devices it can take up to three seconds for a hard disk drive to spin up and start the enumeration process. Because this is the only way of detecting that the hardware is attached, we have to wait for a minimum amount of time (two seconds).

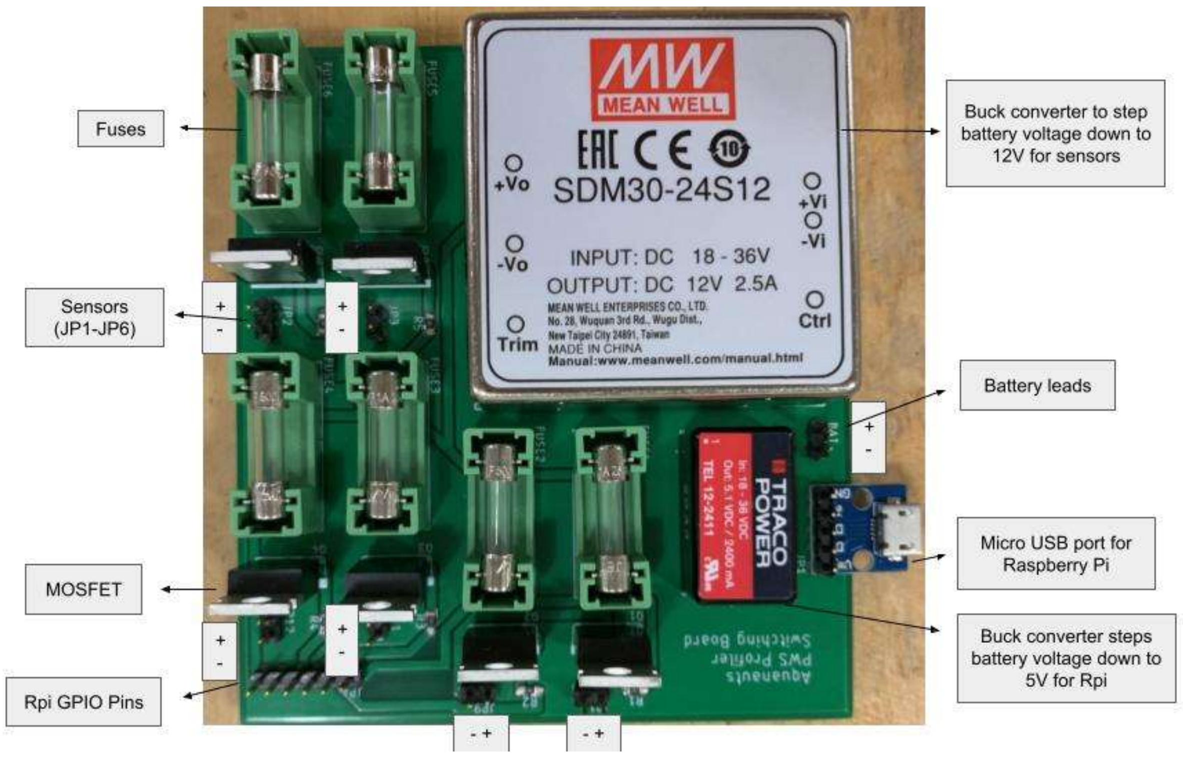
b. This issue can be resolved using the following. **Note: this change must initially be made on the SD Card when configuring boot options, and is also recommended to be made on the SSD itself.** More information regarding boot sequence and timing can be found at the following link: [Boot sequence - Raspberry Pi Documentation]

If the device fails to respond after this maximum timeout, it is possible to increase the timeout to five seconds using `program_usb_boot_timeout=1` in `config.txt`.

3. Establish connection between all parts:
   a. The SSD will connect to the Pi's USB hub
   b. The cell kit will sit atop the Pi, connected through GP/IO interface
   c. The circuit board will connect through pins to the Pi, and the pins used are up to you (See appendix for Sixfab Cellular HAT pinout, and section 3.4 for more information).
   d. The Pi must be connected to power
   e. The cell kit should be powered separately through the micro-USB connector on the side.
4. Power up the Pi.
   a. The Green ACT light next to the red power light should flash lightly, indicating that the SSD is being read
      i. In case of failure to read from SSD, flash a version of the Raspbian OS onto SD Card (using Raspberry Pi Imager - open source and online) and insert the SD card into the Pi - this will bypass boot from USB. More information can be found here: [Installing operating system images - Raspberry Pi Documentation]

   b. Connect through HDMI to a monitor to have access to command line and code within the Pi
   c. Connect a keyboard and mouse to the Pi using the USB hub
5. Run necessary code or scripts from the Pi
   a. You can create and run scripts from command line using Ubuntu commands
   b. Scripts can be scheduled to run using linux cron jobs (See section 3.4)
   c. In the future, SSH'ing into the Pi will allow for use without the HDMI port, in a process outlined here: https://www.raspberrypi.org/documentation/remote-access/ssh/

2.2 PCB Instructions


Fuses
Sensors (JP1-JP6)
MOSFET
Rpi GPIO Pins
Buck converter to step battery voltage down to 12V for sensors
Battery leads
Micro USB port for Raspberry Pi
Buck converter steps battery voltage down to 5V for Rpi
+
-
- +
MEAN WELL
SDM30-24S12
INPUT: DC 18 - 36V
OUTPUT: DC 12V 2.5A
TRACO POWER
Aquanauts PWS Profiler Switching Board

The figure above shows what all the components on the PCB are and their polarities for connecting parts.

1. Connect battery leads to the PCB following the polarity on the labeled figure above.

2. Use a micro USB to micro USB cable to connect the RPi to the PCB to receive power
3. Connect up to six sensors to their pin headers on the PCB. Ensure the positive and negative leads are attached following the labels on the above image.
4. Select RPi GPIO pins and connect them to the corresponding sensor’s pin on the pin header on the PCB.

3. How to Use

This section covers basic commands for sending files to a remote server with an FTP connection. The system originally designed by the Aquanauts uses the Sixfab cell kit with the Telit LE910C1-NF chip. **However, it is recommended for the user to acquire a cell kit with the Quectel EC25EC21 chip in the future (discussed in section 3.3).** The commands for sending data are similar and are discussed in their respective sections, 3.1 and 3.2. For more information on how to use Sixfabs hardware and software, visit the following link: https://docs.sixfab.com/.

3.1 Telit FTP Commands

The following commands can be used for sending a single file with the Telit module. See the references section for more details if this does not suit your needs [1].

1) AT+CGDCONT=<cid>,<PDP_type>,<APN>
    - <cid> integer that sets the context ID, which can be 1-16
    - <PDP_type> string that sets the Packet Data Protocol type can be one of the following:
        + "IP" - IPv4
        + "PPP" - Point to Point
        + "IPv6" - IPv6
    - <APN> string of the access point name
        + This is usually the name or IP of the nearest cell tower
2) AT#PDPAUTH=<cid>,<auth_type>,<username>,<password>
    - <cid> is an integer of the context ID 1-16
    - <auth_type> is an integer that represents the authentication type:
        + 0: no authentication (default)

        + 1: PAP
        + 2: CHAP
    - <username> and <password> are strings of the credentials given to you by your internet provider (not needed for authentication type 0)

3) AT#GPRS=1
    - GPRS context must be attached before opening an FTP connection
        + 0 would deactivate this
4) AT+CGACT=<state>,<cid>
    - <state> deactivates or activates the PDP context with 0 or 1, respectively
    - <cid> is an integer of the context ID 1-16
5) AT#FTPOPEN=<server:port>,<username>,<password>,<mode>
    - This command opens the FTP connection
    - <server:port> is of type string
        + The server can be in IPv4 or IPv6 format
        + The port number is 21 by default
    - <mode> can be 0 or 1, which are active or passive, respectively
        + It is recommended to provide the port number and to use active mode
6) AT#FTPCFG=<tout>,<IPPignoring>,<FTPSEn>,<FTPext>
    - <tout> is the time-out interval in 100 ms units (default is 100 ms and max is 5000 ms)
    - <IPPignoring> disables or enables the use of the private IP sent by the server with 0 or 1, respectively
        + It is recommended to set this to enable this to 1
    - <FTPSEn> disables or enables FTPS for security with 0 or 1, respectively
    - <FTPext> -- this is recommended to be set to 0 (see references for more details [1])
7) AT#FTPPUT=<filename>,<connMode>

- This command sends a file
- <filename> is a string, which can be no more than 200 characters long
- <connMode> can be 0 or 1, which are online or command, respectively
    + It is recommended to set this to 0 for online mode

8) AT#FTPTYPE=<type>
    - <type> sets the file transfer type to binary or ascii with 0 or 1, respectively

9) AT#FTPCLOSE
    - This command closes the FTP connection

***Note: Please reference the following link for how to set up a connection to your local cellular network based on your carrier: Data Call – AT commands to set up GPRS/EDGE/UMTS/LTE data call | M2MSupport.net

3.2 Quectel FTP Commands

The following commands are all you need for sending a single file with the Quectel module. This walkthrough will strictly use FTP rather than FTPS. See the references for more details if this does not suit your needs, or if you would like to use FTPS instead [2][3][4].

1) AT+QICSGP=<contextID>,<context_type>,<APN>,<username>,<password>,<authentication
>
    - <contextID> integer from 1-16
    - <contect_type> can be set to 1 or 2, which is IPv4 or IPv6, respectively
    - <APN> string of the access point name
        + This is usually the name or IP of the nearest cell tower
    - <username> and <password> are strings of your cellular account credentials

- <authentication> integer to determine the authentication method
    + 0: NONE
    + 1: PAP
    + 2: CHAP
    + 3: PAP or CHAP

2) AT+QIACT=<contextID>
    - Activates the context created in the last step
3) AT+QFTPCFG="contextid",<contextID>
    - Configures and uses the context
4) AT+QFTPCFG="account",<username>,<password>
    - <username> and <password> are strings used to log into the FTP server
5) AT+QFTPCFG="filetype",<filetype>
    - <filetype> integer that determines the filetype:
        + 0: binary
        + 1: ascii
6) AT+QFTPCFG="transmode",<transmode>
    - <transmode> integer that determines the filetype:
        + 0: active mode (recommended for this project)
        + 1: passive mode
7) AT+QFTPCFG="rsptimeout",<timeout>
    - <timeout> integer that determines the timeout interval in seconds (can be 20-180)
8) AT+QFTPOPEN=<hostname>,<port>
    - This command opens the connection to the FTP server
    - <hostname> and <port> are both strings, and are related to the FTP server itself
9) AT+QFUPL=<filename>

- Uploads the desired file

10) AT+QFTPCLOSE

    - Closes the connection

***Note: Please reference the following link for how to set up a connection to your local cellular network based on your carrier: Data Call – AT commands to set up GPRS/EDGE/UMTS/LTE data call | M2MSupport.net

3.3 Benefits to Using the Quectel Chip

It is recommended that the Sixfab Cellular Kit is specified with the Quectel EC25 chip. This is an option provided by Sixfab when purchasing the Sixfab 4G/LTE Cellular Kit. This is due to the fact that the Telit LE910C1-NF chip is not compatible with AT commands via UART, while the Quectel EC25 chip is compatible with AT commands via UART. The Sixfab Cellular Kit will require its own power source to avoid causing low power warnings from the Raspberry Pi, thus occupying the USB port on the cellular kit and eliminating the possibility of sending AT commands from the Raspberry Pi to the Sixfab Cellular HAT via USB.

3.4 Turning Sensors On and Off

To turn sensors on and off: (Note: all script examples are in Python syntax)

1) Individually connect each MOSFET on the PCB to any available GPIO pins on the Raspberry Pi. The only pins in use are GPIO14 (pin 8), GPIO15 (pin 10), GPIO6 (pin 31), GPIO13 (pin 33), GPIO19 (pin 35), and GPIO26 (pin 37) all occupied by the Sixfab Cellular HAT. Note: see the appendix for a pinout of the Sixfab Cellular HAT.
2) To set up pins for voltage output, use the following example:

a) GPIO.setmode() accepts two parameters: GPIO.BCM, and GPIO.BOARD. GPIO.BCM will use GPIO numbering (GPIO23 is pin 23), while GPIO.BOARD will use board numbering (GPIO23 is pin 16).

b) GPIO.setup defines whether the pin is an input pin or an output pin. For this use case, the pins will always be set up as output pins using the GPIO.OUT parameter. There needs to be one GPIO.setup command per pin. Example:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM) #use BCM numbering
GPIO.setup(16, GPIO.OUT) #set up GPIO16 as output pin
```

3) To turn individual pins on and off:

a) The GPIO.output() command accepts two parameters: a pin number, and 1 (high) or 0 (low). Example:

```
GPIO.output(16,1) #sets GPIO16 to high using BCM numbering
```

The information above can be used to formulate a script for turning individual pins on and off.

Example script to set all pins high (all connected sensors ON):

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM) #use BCM numbering
GPIO.setup(23, GPIO.OUT) #set up GPIO23 as output pin
GPIO.setup(24, GPIO.OUT)
GPIO.setup(25, GPIO.OUT)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(22, GPIO.OUT)
GPIO.setup(27, GPIO.OUT)
GPIO.output(23,1) #set GPIO23 high
GPIO.output(24,1)
GPIO.output(25,1)
GPIO.output(17,1)
GPIO.output(22,1)
GPIO.output(27,1)
```

Example script to set all pins low (all connected sensors OFF):

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM) #use BCM numbering
GPIO.setup(23, GPIO.OUT) #set up GPIO23 as output pin
GPIO.setup(24, GPIO.OUT)
GPIO.setup(25, GPIO.OUT)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(22, GPIO.OUT)
GPIO.setup(27, GPIO.OUT)
GPIO.output(23,0) #set GPIO23 low
GPIO.output(24,0)
GPIO.output(25,0)
GPIO.output(17,0)
GPIO.output(22,0)
GPIO.output(27,0)
```

4) To schedule this automatically, the simplest and most straightforward method is to execute your script inside of a cron job. A cron job is an automated task that will execute regularly on a defined schedule. More information about cron jobs can be found at the following link: How To Schedule Python Scripts As Cron Jobs With Crontab (Mac/Linux) | Python Engineer (python-engineer.com)

4. Monitoring the Sixfab Cell Kit

Sixfab offers an easy to use interface to interact with the SIM card of the device. This can be located here: https://connect.sixfab.com/#/auth/login. The current login is associated with a student's e-mail and information, where the login information is:

ID: 4873_project

Password: drwestiscool

On this site, the usage can be monitored, billing occurs for the SIM card, and you can perform remote maintenance on the SIM card. The instructions are easy to follow and provide an ease of access that's unmatched. For more information, visit the following link: https://docs.sixfab.com/docs/sixfab-core-introduction.

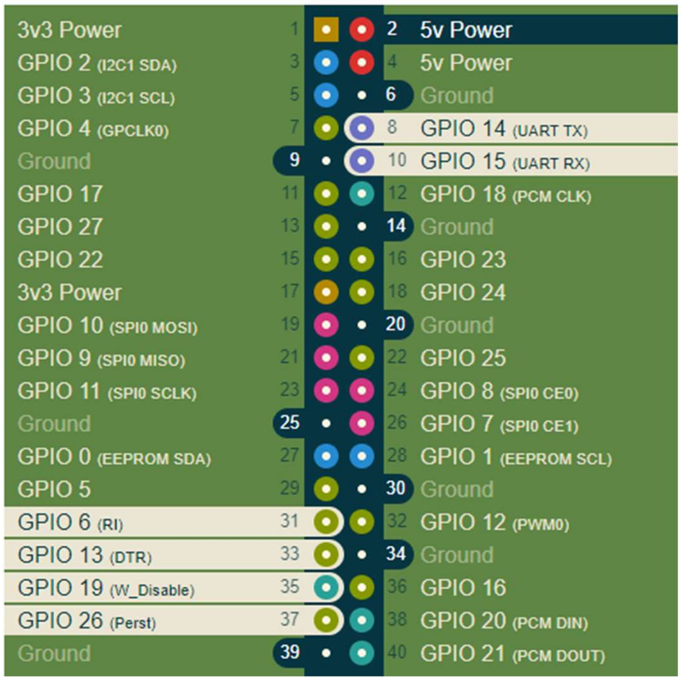
5. References

1. *Telit_LE910Cx_AT_Commands_Reference_Guide_r8.pdf*. Accessible online at: https://sixfab.com/wp-content/uploads/2020/08/Telit_LE910Cx_AT_Commands_Reference_Guide_r8.pdf.
2. QuectelEC2x26EG9x26EM05FTPSATCommandsManualV10.116697075.pdf. Accessible online at: https://usermanual.wiki/Document/QuectelEC2x26EG9x26EM05FTPSATCommandsManualV10.116697075/view.
3. *Quectel_EC2xEG9xEM05_TCPIP_AT_Commands_Manual_V1.0.pdf*. Accessible online at: https://sixfab.com/wp-content/uploads/2018/09/Quectel_EC2xEG9xEM05_TCPIP_AT_Commands_Manual_V1.0.pdf.
4. *Quectel_EC20_FILE_AT_Commands_Manual.pdf*. Accessible online at: https://www.cika.com/soporte/Information/GSMmodules/Quectel/EC20/Quectel_EC20_FILE_AT_Commands_Manual.pdf.

Appendix

Sensors	Min Voltage	Max Voltage	Min Current (mA)	Max Current (mA)
SBE 16plus V2	9	28	N/A	N/A
SBE 43	6.5	24	2.5	9.23
SUNA Nitrate Sensor	8	18	416.67	937.5
Echo Fluorometer	7	15	60	140

Sixfab Cellular HAT Pinout


3v3 Power 1 2 5v Power
GPIO 2 (I2C1 SDA) 3 4 5v Power
GPIO 3 (I2C1 SCL) 5 6 Ground
GPIO 4 (GPCLK0) 7 8 GPIO 14 (UART TX)
Ground 9 10 GPIO 15 (UART RX)
GPIO 17 11 12 GPIO 18 (PCM CLK)
GPIO 27 13 14 Ground
GPIO 22 15 16 GPIO 23
3v3 Power 17 18 GPIO 24
GPIO 10 (SPI0 MOSI) 19 20 Ground
GPIO 9 (SPI0 MISO) 21 22 GPIO 25
GPIO 11 (SPI0 SCLK) 23 24 GPIO 8 (SPI0 CE0)
Ground 25 26 GPIO 7 (SPI0 CE1)
GPIO 0 (EEPROM SDA) 27 28 GPIO 1 (EEPROM SCL)
GPIO 5 29 30 Ground
GPIO 6 (RI) 31 32 GPIO 12 (PWM0)
GPIO 13 (DTR) 33 34 Ground
GPIO 19 (W_Disable) 35 36 GPIO 16
GPIO 26 (Perst) 37 38 GPIO 20 (PCM DIN)
Ground 39 40 GPIO 21 (PCM DOUT)