REMOTE MONITORING PROGRAM
REMOTE SENSOR
PROJECT OUTPOUR
FIRMWARE SPECIFICATION

Version 2.8
08/07/2015

`

VERSION HISTORY

Version	Implemented	Revision	Approved	Reason
1.0	John Vinyard	05/21/2014	Robert Lee	Initial Firmware Specification draft
1.1	John Vinyard	05/30/2014		Added data packet sample
1.2	John Vinyard	07/08/2014		Learning algorithm spec
1.3	John Vinyard	8/08/2014		Red flag packet updated to be more like weekly packet; calendar and clock updated to include an OTA that updates a device with it's local time and date
1.4	John Vinyard	9/09/2014		Add detail on frequency of Pad 5 check in different states; Start Remote Update section
1.5	John Vinyard	10/08/2014		Remote Update definitions added
2.0	John Vinyard	10/21/2014		• Changed activate test idle time from 60s to 30s • Added once monthly test for activation via transmission • Changed clock OTA update to blank out all previous storage • Add to red flag reset that daily volumes must be >12.5% of pre-90 day averages
2.1	John Vinyard	12/19/2014		Message updated (coming)
2.2	John Vinyard	01/15/2015		OTA Reply structure detailed with life cycle assumptions listed
2.3	John Vinyard	1/16/2015		JSON packets added, memory map updated
2.4	John Vinyard	2/6/2015		Updated Packets and OTA based on testing
2.5	Lauren Meleney			Changed data packet specifications to daily only
2.6	D. Laone	4/12/2015		Add new message structure information sections
2.7	Robert Lee	4/20/2015		
2.8	Robert Lee	08/07/2015		OTA reply expansion, edits to documented data structure, updates to feature descriptions

Overview

The Outpour firmware uses capacitive sensors to determine the water level in the head of a small-bore Afridev pump. It senses and records water levels and estimates total liter flow. Once a week it connects to the 2G GSM network and uploads data for the amount of water flow for each hour of that week. In the case of unexpected low usage, it uploads data on that same day instead of at the end of the week. This document outlines the structure of the software system and the subsystems that make it up.

System Architecture

The system is architected to wake on an interrupt each second to handle sensing, storage, and transmission, then to sleep for the remainder. Each module is guaranteed to complete any task in under a second, or return to the main thread and store enough information to continue in memory. Unit testing will ensure that the longest path of each module summed together is under 1 second.

Scheduling

The firmware is divided into three separate modules with limited communication between each. The modules are:

1. Water Sensing
2. Storage
3. Modem Communication

The main loop runs the water sensing module which is responsible for activating the capacitive sensors and estimating the water flow. It then runs the storage module every second, simplifying it's internal timekeeping. Finally the modem FSM is run, initiating GSM transactions and recording transmission status.

Each module has a unit testing environment. Modules are written to interface with a HAL that replaces the MSP430-specific code with generic abstractions for the hardware. Test plans for each module are in a separate document.

Water Sensing

This module activates the sensors and converts the raw readings into an estimated mL. It determines how long to be idle between readings to save power. It activates the unit on installation.

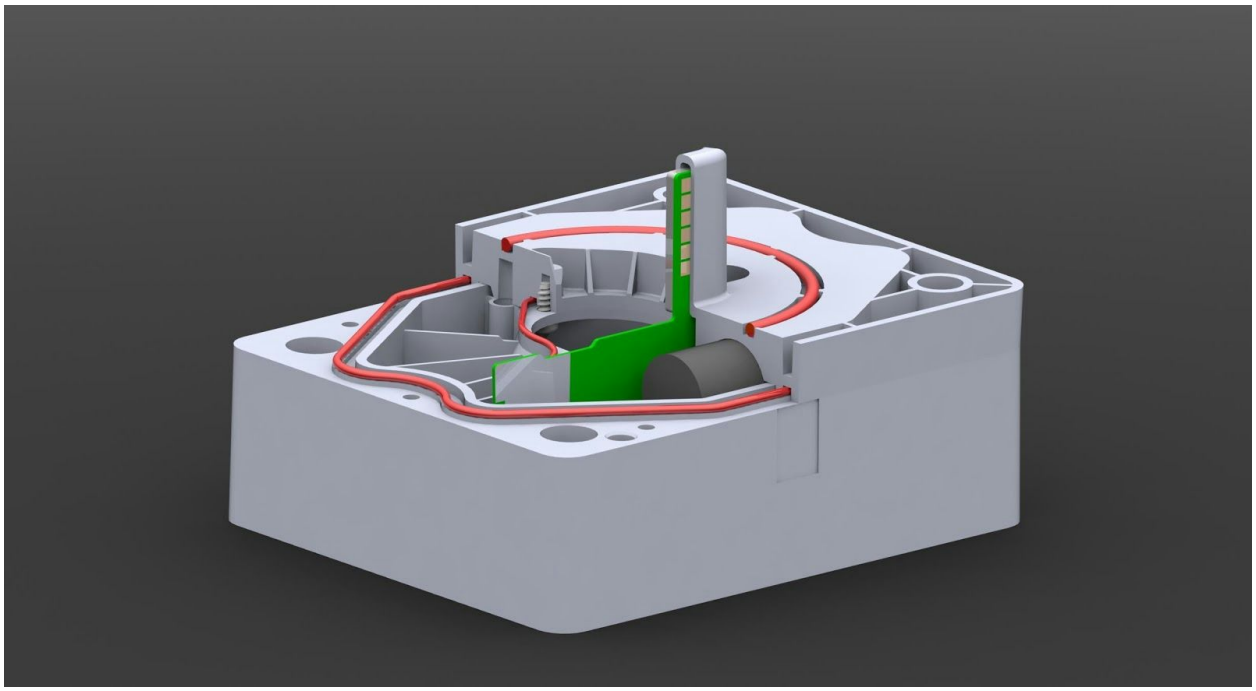


Figure A: Pad5 is the lowest. Pad0 is at the top tip of the finger.

Each sensor's reading is compared to the past hour's maximum value seen. If the delta is greater than the predetermined threshold, the pad is considered "submerged". The highest "submerged" pad is used to estimate that second's milliliters. If hourly stored estimates are unavailable, daily usage can be calculated from the sensor statistics alone.

The module does some bookkeeping on sensor statistics and exits back to the scheduler. It tracks minimum and maximum pad values along with submerged seconds. These are reset with a function call from the Storage module indicating it has recorded the values for the day.

Activation FSM

1. Battery plugged in
2. Check for water at low frequency rate (1/60 Hz)
3. If water sensed, check for water at high frequency rate (1 Hz); once no water is sensed for 5 minutes, go back to #2.
4. If daily flow = 200L, then Activate; else, go back to #2.

In the “Installed” state, after 300 seconds of no sensors being submerged, the delay between sense measurements is increased. The micro will still wake up every second for the modem and storage modules, but the water sensing code is not invoked. The low-frequency ‘sleep’ checks the pads every 60 seconds.

Also, monthly check-in to see if there is a Message that instructs unit to go into Activation (if it hasn’t been activated with a 200L day).

Data Storage Information

Hourly liter information is stored in fixed point 11.5. 11 bits of liters, 5 bits of sub-liter precision. The water sensing module has results in terms of mL. The storage module samples this every second and keeps a running total of mL for the minute in an unsigned int. The running total for the hour is stored in an unsigned long (32 bits). At the end of the hour, the total is shifted right 5 bits (/32) and written to the nonvolatile flash. At the end of the day, all hourly liters are summed up and shifted right another 5 bits to store total liters as an unsigned int. The maximum the pump can do is ~35,000, the maximum storage is ~65,000.

Maximum flow: 24L/minute
Maximum liters per minute: 24L
Maximum storage in uint16_t: 65536 mL
Maximum liters per hour: 1440
Maximum storage in uint32_t: 4294967296 mL

Calendar and Clock

The main calendar is not used in the storage FSM. The beginning date and time are stored in the daily packet in GMT, but the storage FSM keeps time in 168-hour chunks representing an idealized week/month pattern. Remote updates will allow this window to be adjusted independent of the GMT time: e.g. if the unit is installed in Ethiopia (GMT+3) an OTA packet will be sent to the unit to adjust the storage FSM’s clock ahead by 3 hours. This will also enable fine-grained adjustments if the clock or crystal begins to drift.

Red Flag

The storage module is also responsible for generating “Red Flag” events. This occurs when a pump sees unexpectedly low flow rate over a single day and reports out of band. After a red flag event is reported, regular weekly transmissions resume without further low usage being reported.

During the first 4 weeks after installation, the firmware records the total daily liters. At the end of 4 weeks, it takes the average of the 4 values for each day of the week and defines a weekly map.

E.g. if the first 4 weeks looked like this:

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
3276	2654	3356	3275	3176	2985	2871
3158	2713	3245	3182	3008	3060	2994
3314	2694	3158	3074	3157	2956	3012
3214	2718	3215	3013	3089	3021	2852

The weekly map would be:
{3240, 2694, 3243, 3136, 3107, 3005, 2932}

The map is updated each week to reflect the average of the previous four weeks.
After the four week learning period: at the end of each day, the total liters is compared to the previous day's total and the weekly map value. If the total is less than 50% of the map value and the map value is at least 200 liters, a Red Flag event is generated. If no Red Flag is generated, the day's value is incorporated with the weekly map value, weighted 25% new value / 75% old weekly map value, and the new value is placed in the weekly map.

The Red Flag is sent at the end of the day when that threshold is hit. The packet is the same size and format as the Weekly packet, with the days that have not happened yet taking on the default value of 0xFF for all bytes.

Red Flag packets arriving on days 1-6 of the week can be detected by looking for the default values present in later days of the week. If a Red Flag occurs on day 7 of the week, all of the daily packets will have initialized values and this check is not sufficient. This case can be detected by checking that the Red Flag field of the weekly packet is written to 0x01 and confirmed by checking the Red Flag of each day's logs to see days 1-6 as 0x00 and day 7 written as 0x01.

The Red Flag is reset when either flow resumes to 75% of the average or 90 days pass and flow resumes to at least 12.5% of the average. If the 90 day limit resets it, the unit will wipe out the map and assume new usage patterns have been set.

Unit Life Cycle

The expected life cycle of the units is as follows:

1. Battery plugged in
2. FA Message goes out, used to indicate that the unit can be shipped to the operator
3. FA Message data parsed by C:W server to determine time delta
4. OTA GMT Update queued up for unit
5. FA packet sent again, unit downloads GMT Update packet
6. Advances internal clock to correct GMT time
7. OTA Reply to indicate new time
8. Unit shipped to destination
9. Unit logged on installation, Storage Offset calculated
10. Storage Offset OTA sent
11. Unit activates and connects
12. Storage OTA downloaded
13. Unis is Activated, has correct GMT time, correct Storage offset, and begins normal operation

Transmit Message Packet Format Description (MSP430->Cloud)

This section identifies the format of each message sent by the MSP430 to the server.

Transmit Message Summary

msgId	Name	Description
0x00	Final Assembly	Mechanism to get time at manufacturing
0x01	Daily Log	Water data for one 24 hour period (1 day)
0x02	Weekly Log	NOT USED ANYMORE
0x03	OTA Reply	Let server know that the OTA message was received
0x04	Retry	Let server know that a message retry was performed
0x05	Monthly Check-In	Unit check-in with server on a monthly basis (every 28 days)

Common Packet Header Structure

Each transmitted packet has a common header format. It consists of 16 bytes. Specific and custom data to each message type follows the common header structure.

Note: For all 16 bit data parameters, the most significant byte is sent first.

Byte Offset	Field	Description	Size	Note
0	msgType	Specifies the message type (1=data)	1 byte	**value = 0x01**
1	msgId	Specifies the message identifier	1 byte	
2	productId	Product Identifier	1 byte	
3	gmtSecond	GMT second	1 byte	Binary
4	gmtMinute	GMT minute	1 byte	Binary
5	gmtHour	GMT hour (24 hour)	1 byte	Binary
6	gmtDay	GMT day	1 byte	Binary
7	gmtMonth	GMT month	1 byte	Binary
8	gmtYear	GMT year (tens only, i.e. 15, 16)	1 byte	Binary
9	fwVersionMajor	x of x.y version number	1 byte	
10	fwVersionMinor	y of x.y version number	1 byte	
11-12	dayCount	Count of days since activation	2 bytes	MSB first
13-15	reserved		3 bytes	

Final Assembly Message (msgId = 0x00)

Message Description

As part of the Final Assembly test, the unit sends a message containing its internal GMT clock values. The message is sent twice. The first is sent 3 minutes after boot. The second is sent 6 minutes after boot. The purpose in sending it twice is to allow the server to send the OTA GMT Clock Update message containing the current GMT time to the unit so that it can be processed as part of sending the second Final Assembly message.

Packet Structure

Byte Offset	Field	Description	Size	Note
0	msgType	Specifies the message type (1=data)	1 byte	**value = 0x01**
1	msgId	Specifies the message identifier	1 byte	**value = 0x00**
2	productId	Product Identifier	1 byte	
3	gmtSecond	GMT second	1 byte	Binary
4	gmtMinute	GMT minute	1 byte	Binary
5	gmtHour	GMT hour (24 hour)	1 byte	Binary
6	gmtDay	GMT day	1 byte	Binary
7	gmtMonth	GMT month	1 byte	Binary
8	gmtYear	GMT year (tens only, i.e. 15, 16)	1 byte	Binary
9	fwVersionMajor	x of x.y version number	1 byte	
10	fwVersionMinor	y of x.y version number	1 byte	
11-12	dayCount	Count of days since activation	2 bytes	MSB first
13-15	reserved		3 bytes	

Water Daily Log Message (msgId = 0x01)

Message Description

Water data for one day is transmitted in a daily log packet structure. Each daily log packet of the current week is transmitted as a separate message at the end of the week. Hence, 7 daily log messages will be transmitted at the end of each week.

Packet Structure

Byte Offset	Field	Description	Size	Note
	start packet header			
0	msgType	Specifies the message type (1=data)	1 byte	**value = 0x01**
1	msgId	Specifies the message identifier	1 byte	**value = 0x01**
2	productId	Product Identifier	1 byte	
3	gmtSecond	GMT second at start of day's recording	1 byte	Binary
4	gmtMinute	GMT minute at start of day's recording	1 byte	Binary
5	gmtHour	GMT hour at start of day's recording (24 hour)	1 byte	Binary
6	gmtDay	GMT day at start of day's recording	1 byte	Binary
7	gmtMonth	GMT month at start of day's recording	1 byte	Binary
8	gmtYear	GMT year at start of day's recording (tens portion only, i.e. 15, 16, etc)	1 byte	Binary
9	fwVersionMajor	x of x.y version number	1 byte	
10	fwVersionMinor	y of x.y version number	1 byte	
11-12	dayCount	Count of days since activation	2 bytes	MSB first
13-15	reserved		4 bytes	
	end packet header (total header bytes: 16)			
	start packet data			
16-63	liters	Per-hour count of volume	[24] * 2 bytes = 48 bytes	

64-75	padMax	Per-sensor maximum value	[6] * 2 bytes = 12 bytes	
76-87	padMin	Per-sensor minimum value	[6] * 2 bytes = 12 bytes	
88-99	padSubmerged	Per-sensor submerged count	[6] * 2 bytes = 12 bytes	
100-101	comparedAverage	Red flag average before this day	2 bytes	
102-103	unknowns	Count of unknown “splash” readings	2 bytes	
104	redFlag	Red Flag on current day’s liter count	1 byte	
105-127	reserved	Expansion for future	23 bytes	
	end packet data (total data bytes: 112)			
	End Packet (total packet bytes: 128)			

OTA Reply Message (msgId = 0x03)

Message Description

After each successful OTA message is received by the MSP430, it sends a reply message back to the server to confirm that the message was received and the internal state of the unit updated accordingly. Each OTA message received by the MSP430 contains the message identifier opcode and a two byte message ID. The OTA opcode and message ID are echoed back to the server in the OTA Reply Message to tell the server that the OTA message was received and processed.

Packet Structure

Byte Offset	Field	Description	Size	Note
0	msgType	Specifies the message type (1=data)	1 byte	**value = 0x01**
1	msgId	Specifies the message identifier	1 byte	**value = 0x03**
2	productId	Product Identifier	1 byte	
3	gmtSecond	GMT second	1 byte	Binary
4	gmtMinute	GMT minute	1 byte	Binary
5	gmtHour	GMT hour (24 hour)	1 byte	Binary
6	gmtDay	GMT day	1 byte	Binary
7	gmtMonth	GMT month	1 byte	Binary
8	gmtYear	GMT year (tens only, i.e. 15, 16)	1 byte	Binary
9	fwVersionMajor	x of x.y version number	1 byte	
10	fwVersionMinor	y of x.y version number	1 byte	
11-12	dayCount	Count of days since activation	2 bytes	MSB first
13-15	reserved		3 bytes	
16	otaMessageId	The OTA Message ID	1 byte	
17-18	msgNumber	Echo message number for message replying to	2 bytes	

19	replyMsg	Reply data	29 bytes	

Retry Message (msgId = 0x04)

Message Description

If a daily log message or monthly check-in message transmission fails because the modem can not connect to the network, then a re-transmission will be attempted in 12 hours from the original transmission. The modem stores all data messages internally if they were not successfully transmitted. In order to get the modem to send out these stored messages, it must "kicked" by sending it a new data type message. To that end, a retry message is used to kick the modem to send any previously stored messages.

Packet Structure

Byte Offset	Field	Description	Size	Note
0	msgType	Specifies the message type (1=data)	1 byte	**value = 0x01**
1	msgId	Specifies the message identifier	1 byte	**value = 0x04**
2	productId	Product Identifier	1 byte	
3	gmtSecond	GMT second	1 byte	Binary
4	gmtMinute	GMT minute	1 byte	Binary
5	gmtHour	GMT hour (24 hour)	1 byte	Binary
6	gmtDay	GMT day	1 byte	Binary
7	gmtMonth	GMT month	1 byte	Binary
8	gmtYear	GMT year (tens only, i.e. 15, 16)	1 byte	Binary
9	fwVersionMajor	x of x.y version number	1 byte	
10	fwVersionMinor	y of x.y version number	1 byte	
11-12	dayCount	Count of days since activation	2 bytes	MSB first
13-15	reserved		3 bytes	

Monthly Check-in Message (Type = 0x05)

Message Description

As a method to ensure that the unit will always communicate to the server at least once a month, the monthly check-in message is used. This allows the unit to download any OTA messages from the server even if the unit is not yet activated. Note that the monthly check-in message is based on a storage month (28 days). The unit will not send the monthly check-in message if it is activated and sending the water data log packets.

Packet Structure

Byte Offset	Field	Description	Size	Note
0	msgType	Specifies the message type (1=data)	1 byte	**value = 0x01**
1	msgId	Specifies the message identifier	1 byte	**value = 0x05**
2	productId	Product Identifier	1 byte	
3	gmtSecond	GMT second	1 byte	Binary
4	gmtMinute	GMT minute	1 byte	Binary
5	gmtHour	GMT hour (24 hour)	1 byte	Binary
6	gmtDay	GMT day	1 byte	Binary
7	gmtMonth	GMT month	1 byte	Binary
8	gmtYear	GMT year (tens only, i.e. 15, 16)	1 byte	Binary
9	fwVersionMajor	x of x.y version number	1 byte	
10	fwVersionMinor	y of x.y version number	1 byte	
11-12	dayCount	Count of days since activation	2 bytes	MSB first
13-15	reserved		3 bytes	

Receive Message Packet Format Description (Cloud->MSP430)

After any successful message transmission, the firmware must check the modem for remote messages. These are messages sent by the server to the unit. These are also referred to as OTA (over the air) messages.

Receive Message Summary

Opcode	Name	Action
0x01	GMT Clock Update	Advance GMT Clock by the specified time. Used in Final Assembly test
0x02	Storage Clock Alignment	Start Storage Window at the specified GMT time on the next day
0x03	Reset Data	Erase all logged data, de-activate
0x04	Reset Red Flag	Reset global Red Flag event
0x05	Transmissions On ("Activate Device")	Activate device. This enables the device to send the water log message.
0x06	Transmissions Off ("Deactivate Device")	De-activate device.
0x07	Update Constants	Update pad thresholds and flow constants
0x08	Reset Device	Forces a microprocessor reset

Common Packet Header Structure

Each OTA packet sent by the server has a common header format. It consists of 3 bytes. Specific/custom data to each receive message type follows the common header structure.

Note on the msgNumber field:
It is intended that the msgNumber field value is a counter that is incremented for each OTA packet sent to a device. That way each message will have a unique identification number.

Byte Offset	Field	Description	Size	Note
0	opcode	Specifies the message type (1=data)	1 byte	
1-2	msgNumber	Unique identification number that will be echoed back to server in OTA reply	2 bytes	

GMT Clock Update (opcode=0x01)

Message Description

The Final Assembly test consists of the packaged unit sending out it's internal clock. The server calculates the difference between the reported clock and the GSM timestamp in the packet, then prepares this message. It can also be sent at a later time if the clock is observed to be drifting. Upon receiving this message, the unit will advance it's internal clock ahead by the given amounts.

The example packet below advances the clock by 4 days, 2 hours, **22** minutes, and **51** seconds. The opcode is 0x1, and the message ID is 0x1122.

0x01 0x11 0x22 0x33 0x16 0x02 0x00 0x04

Notes:

- 1 byte of Seconds, 1 byte of Minutes, 1 byte of Hours, and 2 bytes of days allows for advancement of 179 years.
- The time values are sent as binary (not BCD)

Packet Structure

Byte Offset	Field	Description	Size	Note
0	opcode	Specifies the message type (1=data)	1 byte	**value = 0x01**
1-2	msgNumber	Identification number that will be echoed back to server in OTA reply	2 bytes	
3	gmtSecond	seconds to advance GMT time	1 byte	**Binary Value**
4	gmtMinute	minutes to advance GMT time	1 byte	**Binary Value**
5	gmtHour	hours (24 hour) to advance GMT time	1 byte	**Binary Value**
6	gmtDay	days to advance GMT time	1 byte	**Binary Value**

Storage Clock Alignment (opcode=0x02)

Message Description

This message is used to inform the unit to start tracking the next week at the specified GMT time the next day.

Notes:

- Once this message is received by the unit, it will not store any water data until the alignment time is reached
- Receiving this message also blanks any Red Flag information and any previously stored water information
- **Time is encoded in BCD values.**

Example Packets

In these examples, the 02 is the opcode. The 11 22 is the message number (0x1122). The order of the BCD data in the packet is seconds, minutes, hour (one byte each, in BCD format). Hour is assumed to be 24 hour based (there is no concept of AM or PM).

- Start storage when the unit GMT time matches hours=5, minutes=0 and seconds=0
  - 02 11 22 00 00 05
- Start storage when the unit GMT time matches hours=10, minutes=30 and seconds=0
  - 02 11 22 00 30 10
- Start storage when the unit GMT time matches hours=4, minutes=31 and seconds=21
  - 02 11 22 21 31 04

Packet Structure

Byte Offset	Field	Description	Size	Note
0	opcode	Specifies the message type (1=data)	1 byte	**value = 0x02**
1-2	msgNumber	Identification number that will be echoed back to server in OTA reply	2 bytes	
3	gmtSecond	GMT second to match	1 byte	**BCD Value**
4	gmtMinute	GMT minute to match	1 byte	**BCD Value**
5	gmtHour	GMT hour (24 hour) to match	1 byte	**BCD Value**

Reset Data (opcode=0x03)

Message Description

- Resets and erases all stored red flag data
- Resets and erases all stored water data
- De-activates the unit

Packet Structure

Byte Offset	Field	Description	Size	Note
0	opcode	Specifies the message type (1=data)	1 byte	**value = 0x03**
1-2	msgNumber	Identification number that will be echoed back to server in OTA reply	2 bytes	

eg. 03 00 00

Reset Red Flag(opcode=0x04)

Message Description

- Resets and erases all stored red flag data

Packet Structure

Byte Offset	Field	Description	Size	Note
0	opcode	Specifies the msg identifier	1 byte	**value = 0x04**
1-2	msgNumber	Identification number that will be echoed back to server in OTA reply	2 bytes	

Activate Device(opcode=0x05)

Message Description

- Set the unit to activated.

Packet Structure

Byte Offset	Field	Description	Size	Note

0	opcode	Specifies the message type (1=data)	1 byte	**value = 0x05**
1-2	msgNumber	Identification number that will be echoed back to server in OTA reply	2 bytes	

De-Activate Device(opcode=0x06)

Message Description

- Set the unit to de-activated.

Packet Structure

Byte Offset	Field	Description	Size	Note
0	opcode	Specifies the message type (1=data)	1 byte	**value = 0x06**
1-2	msgNumber	Identification number that will be echoed back to server in OTA reply	2 bytes	

Update Constants (opcode=0x07)

Message Description

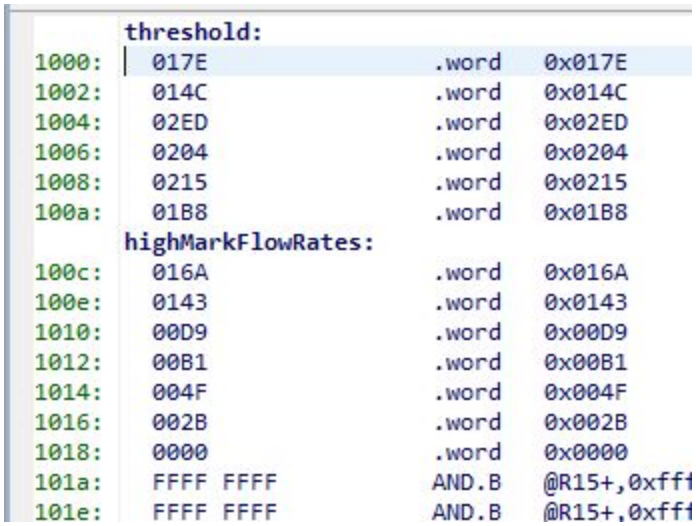
The two sets of critical algorithm constants: (1) per-sensor thresholds and (2) flow rates, can be rewritten from the server. There are a total of twelve constants, with each constant being 16 bits (2 bytes) wide. Six constants for the per-sensor thresholds and six constants for the flow rates. Data in the packet is sent most significant byte first.

Per-Sensor Thresholds:
Thresholds used to identify if a pad is covered with water or not. Used to compare against the capacitive reading differences between max seen (representing air) and current reading.

Flow Rates:
Holds the milliliter per second flow rates values based on pad coverage.

The screenshots below shows an example update:

Original Memory state:

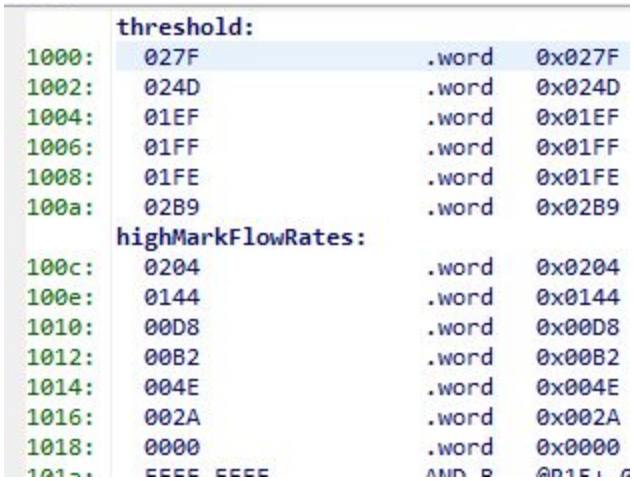


```
        threshold:
1000:   017E                .word   0x017E
1002:   014C                .word   0x014C
1004:   02ED                .word   0x02ED
1006:   0204                .word   0x0204
1008:   0215                .word   0x0215
100a:   01B8                .word   0x01B8
        highMarkFlowRates:
100c:   016A                .word   0x016A
100e:   0143                .word   0x0143
1010:   00D9                .word   0x00D9
1012:   00B1                .word   0x00B1
1014:   004F                .word   0x004F
1016:   002B                .word   0x002B
1018:   0000                .word   0x0000
101a:   FFFF FFFF           AND.B   @R15+,0xfff
101e:   FFFF FFFF           AND.B   @R15+,0xfff
```

Update packet:

07 11 22 02 7f 02 4d 01 ef 01 ff 01 fe 02 b9 02 04 01 44 00 d8 00 b2 00 4e 00 2a 00 00

After the update, the new memory image:



```
        threshold:
1000:   027F                .word   0x027F
1002:   024D                .word   0x024D
1004:   01EF                .word   0x01EF
1006:   01FF                .word   0x01FF
1008:   01FE                .word   0x01FE
100a:   02B9                .word   0x02B9
        highMarkFlowRates:
100c:   0204                .word   0x0204
100e:   0144                .word   0x0144
1010:   00D8                .word   0x00D8
1012:   00B2                .word   0x00B2
1014:   004E                .word   0x004E
1016:   002A                .word   0x002A
1018:   0000                .word   0x0000
```

Packet Structure

Byte Offset	Field	Description	Size	Note
0	opcode	Specifies the message type (1=data)	1 byte	**value = 0x07**
1-2	msgNumber	Identification number that will be echoed back to server in OTA reply	2 bytes	

3-4	pad0Thresh		2 bytes	
5-6	pad1Thresh		2 bytes	
7-8	pad2Thresh		2 bytes	
9-10	pad3Thresh		2 bytes	
11-12	pad4Thresh		2 bytes	
13-14	pad5Thresh		2 bytes	
15-16	flowRatePad0	Flow rate when up through PAD 0 is covered with water	2 bytes	
17-18	flowRatePad1	Flow rate when up through PAD 1 is covered with water	2 bytes	
19-20	flowRatePad2	Flow rate when up through PAD 2 is covered with water	2 bytes	
21-22	flowRatePad3	Flow rate when up through PAD 3 is covered with water	2 bytes	
23-24	flowRatePad4	Flow rate when up through PAD 4 is covered with water	2 bytes	
25-26	flowRatePad5	Flow rate when only PAD 5 is covered with water	2 bytes	
27-28	flowRateNoPads	Flow rate when no pads are covered	2 bytes	

Reset Device (opcode=0x08)

Message Description

- Performs a MSP430 reboot after modem communication is complete (after the OTA reply is sent).

Packet Structure

Byte Offset	Field	Description	Size	Note
0	opcode	Specifies the message type (1=data)	1 byte	**value = 0x08**
1-2	msgNumber	Identification number that will be echoed back to server in OTA reply	2 bytes	

3	keyByte0		1 byte	value = 0xAA
4	keyByte1		1 byte	value = 0x55
5	keyByte2		1 byte	value = 0xCC
6	keyByte3		1 byte	value = 0x33

eg. 08 00 00 AA 55 CC 33

Sending an OTA update via a JSON Object

You'll need to POST a JSON payload containing the message you want to send to the device. You'll need to set HTTP Basic authentication credentials on the request, so we can authenticate you.
The URL is going to be: https://api.bodytrace.com/1/device/{IMEI}/incomingmessage
or http://2.us.data.bodytrace.com:8080/1/device/*{IMEI}*/incomingmessage
where {IMEI} is the IMEI of the device you're sending to (full number number without dashes as seen on barcode)

For example:
http://2.us.data.bodytrace.com:8080/1/device/013777007479619/incomingmessage

The JSON payload will need to have one parameter, data, which needs to hold the message you want sent to the modem base64 encoded.
For example to send the bytes 0xde 0xad 0xbe 0xef to the device, your payload will need to look like this:

```
{"data":"3q2+7w=="}
```

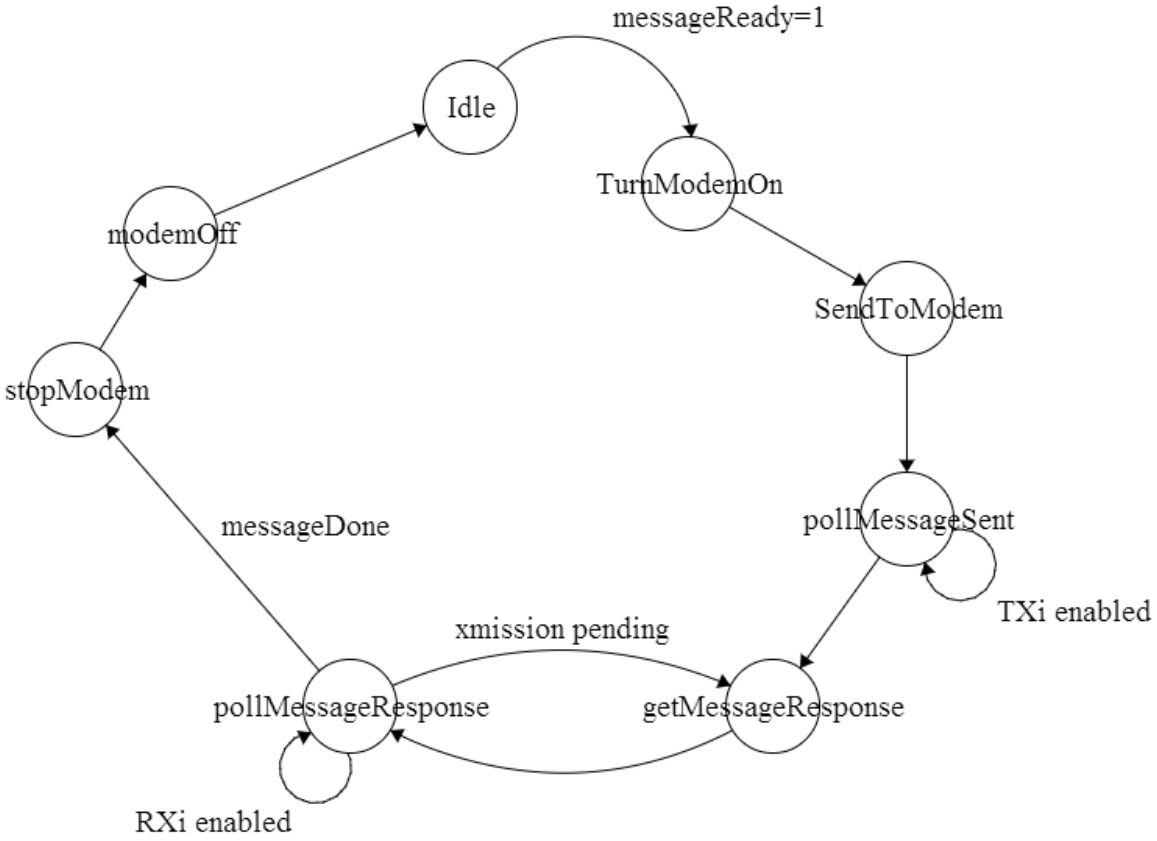
API calls will return an 200 OK or 204 No Content HTTP status code on success or a 4xx-5xx error on failure.

You'll need to POST the message you'd like to send to this URL:
http://2.us.data.bodytrace.com:8080/1/device/<imei>/incomingmessage
where <imei> is the device's IMEI number (numbers only)

Modem Processing Overview

This module handles communication with the BodyTrace modem.

Since modules must complete work in under 1s and modem communications can take much longer, this module is implemented as a FSM. The state diagram:



Any usage of the HW UART block is structured to arrange the necessary buffers, enable the proper interrupt, and return to the main thread. There is a forced transition to a state which tracks the necessary interrupt and handles the results once it is disabled.

Each time a message is ready for the modem, it is guaranteed to transmit. The storage module is responsible for marking a week's worth of data as ready and signalling the information to the modem, at which point this module is responsible for retries and tracking. The weekly data structure contains a field for transmission successful, as well as error codes for unsuccessful transmissions.

Table 5.1 BodyTrace Return Codes

0x00	Initialization after power-on
0x01	Idle (not connected)
0x02-0x03, 0x05-0x79	Transmission in progress

0x04	Idle, previous data transmitted successfully
0x80	Internal error
0x81	Supply voltage too low
0x82	SIM error
0x83	Network error
0x84-0x85	Transmission error
0xa0-0xbf	Provisioning error

The modem transitions to messageDone on 0x04, 0x80-0x85, and 0xa0-0xbf. Other codes cause the modem to loop and wait for another code. If 0xFF is returned multiple times, or the transmission takes longer than 5 minutes, the modem is shut down and the transaction marked to try again later. After a successful message transmission and a return status of 0x04, the firmware must check the modem for remote messages.

Hardware Resources

Table 6.1 Hardware resources

Resource	Usage
UART Module	Electrical communication with GSM modem
Timer A0	Capacitive sensing
Timer A1	Global timekeeping
Watchdog	Capacitive sensing

All timers use the 32khz crystal source. The micro oscillator runs at 1MHz, meaning a maximum of 1 million instructions can run between each timekeeping tick.

Table 6.2 RAM usage

Module	RAM Usage
Modem	0x40 bytes
Water Sense	0x5c bytes
Storage	0x30 bytes
System	0x54 bytes
Remaining	0x47 (14%)

Table 6.3 Non-Volatile Flash layout

Address	Size	Usage
0xc400-0xc800	0x400	Weekly Log #1
0xc800-0xcc00	0x400	Weekly Log #2
0xcc00-0xd000	0x400	Weekly Log #3
0xd000-0xd400	0x400	Weekly Log #4
0x1000-0x101a	0x1a	Threshold/flow rate constants for water sense algorithm

0x1040-0x1048	0x8	Function pointers for modules
0x1080-0x10b8	0x38	First Month daily liters

Appendix A: BodyTrace JSON Packets

Bodytrace delivers data as a JSON object. Examples for each message type is shown on the following pages.

Note: In addition to the data the MSP430 provides, BodyTrace adds the following information to each packet:

- deviceID: The IMEI number of the unit
- ts: Network timestamp, units of milliseconds since epoch
- adc: ??
- rssi: Signal strength indicator
- imei: modem identification number

Daily Log Message Jason Packet

01 01 ff ff ff ff ff ff ff ff ff ff ff ff ff ff 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0a ef 02 e3 03 8e 00 00 00 00 15 67 00 00 00 00 00 00 00 00 00 00 00 00 a6 cf a9 3d 9f c2 ad 05 b5 31 c1 6a a3 1d a4 e5 9b a4 a8 77 b0 47 bb e0 01 fa 03 41 03 82 03 8f 03 a2 03 fb 01 35 00 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
{"waterDailyLog":{"dayCount":65535,"productId":255,"gmtYear":255,"liters":[0,0,0,0,0,0,0,0,0,0,0,2799,739,910,0,0,5479,0,0,0,0,0,0],"unknowns":0,"padSubmerged":[506,833,898,911,930,1019],"gmtMonth":255,"comparedAverage":309,"gmtHour":255,"padMax":[42703,43325,40898,44293,46385,49514],"reserved":"////","gmtDay":255,"gmtSecond":255,"reserved2":"////////////////////////////8=","padMin":[41757,42213,39844,43127,45127,48096],"fwVersionMajor":255,"fwVersionMinor":255,"gmtMinute":255,"redFlag":0}}

Daily Log:

```
{
        "deviceId": 1289600793682,
        "ts": 1399504535103,
        "adc": 4212,
        "rssi": 71,
        “imei”:012896007936825,
        "values": {
                "dailyLog": {
                        “msgType”: 01
                        “msgId”: 01
                        “productId”: 1
                        “gmtSecond”: 36,
                        “gmtMinute”: 53,
                        “gmtHour”: 3,
                        “gmtDay”: 4,
                        “gmtMonth”: 9,
                        “gmtYear”: 15,
                        "fwVersionMajor": 1,
                        "fwVersionMinor": 0,
                        "dayCount": ,
                        “reserved”: ,
                        "liters": [0, 0, 0, 0, 0, 0, 7699, 7717, 7638, 7945, 7796, 7717, 7974,
7776, 7962, 7775, 7898, 7834, 7771, 0, 0, 0, 0, 0],
                        "padMax": [45028, 45509, 46009, 46528, 47015, 47512],
                        "padMin": [43028, 43500, 44035, 44535, 45034, 45512],
```

```
                        "padSubmerged": [8751, 7964, 2718, 6820, 9840, 2277],
                        "comparedAverage": 0,
                        "unknowns": 0,
                        "overflow": 0,
                        “redFlag”: 0
                        “reserved”: [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19, 20 ],
                }
        }
}
```

Final Assembly Jason Packet

```
Final Assembly:
{
        "deviceId": 1289600793682,
        "ts": 1399504535103,
        "adc": 4212,
        "rssi": 71,
        “imei”:012896007936825,
        "values": {
                "productId":0,
                "gmtSecond":49,
                "gmtMinute":2
                "gmtHour":0,
                "gmtDay":1,
                "gmtMonth":1,
                "gmtYear":15,
                "fwVersionMajor":1,
                "fwVersionMinor":3,
                "dayCount":1,
                "reserved":"AAbM",
        }
}
```

The Final Assembly (FA) packet values are decoded by the BodyTrace parser. The Final Assembly packet includes the current GMT time and GMT date that the MSP430 is set to. The MSP430 sends the date and time as binary data (not BCD). The only exception is the Year, which is still in BCD. 4-digit BCD is too complex to undo on the MSP430, so it is not converted to a binary value when sent to the cloud by the MSP430.

Retry Message Jason Packet

```
Retry:{
   "deviceId": 1289600793682,
   "ts": 1399504535103,
   "adc": 4212,
   "rssi": 71,
   "imei": 012896007936825,
   "values": {
      "Retry"
   }
}
```

OTA Reply Jason Packet

eg. 01 03 00 11 01 00 0a 01 0f 01 03 00 01 01 06 cc 05 9a bc
{"otaReply":{"msgNumber":39612,"dayCount":1,"productId":0,"gmtYear":15,"gmtMonth":1,"gmtHour":0,"reserved":"AQbM","gmtDay":10,"gmtSecond":17,"fwVersionMajor":1,"fwVersionMinor":3,"gmtMinute":1,"otaMessageId":5}}

```
OTA Reply:
{
      "deviceId": 1289600793682,
      "ts": 1399504535103,
      "adc": 4212,
      "rssi": 71,
      “imei”:012896007936825,
      "values": {
            "otaMessageId":5
            "msgNumber":39612,
            "productId":0,
            "gmtSecond":17,
```

```
            "gmtMinute":1,
            "gmtHour":0,
            "gmtDay":10,
            "gmtMonth":1,
            "gmtYear":15,
            "fwVersionMajor":1,
            "fwVersionMinor":3,
            "dayCount":1,
            "reserved":"AQbM",
}
```