*TI Designs*

# *Flow Measurement With Reed Switches and BLE*

## TI Designs

TI Designs provide the foundation that you need including methodology, testing and design files to quickly evaluate and customize the system. TI Designs help *you* accelerate your time to market.

## Design Resources

| | |
|---|---|
| TIDM-BLE-REEDMTR | Design Folder |
| CC2650 | Product Folder |
| CC2650EM-71D | Tool Folder |
| SmartRF06 EB | Tool Folder |
| BLE-STACK-2-1 | Tool Folder |
| Code Composer Studio | Tool Folder |
| Sensor Controller Studio | Tool Folder |
| SmartRF Programmer V2 | Tool Folder |

TI E2E™ Community

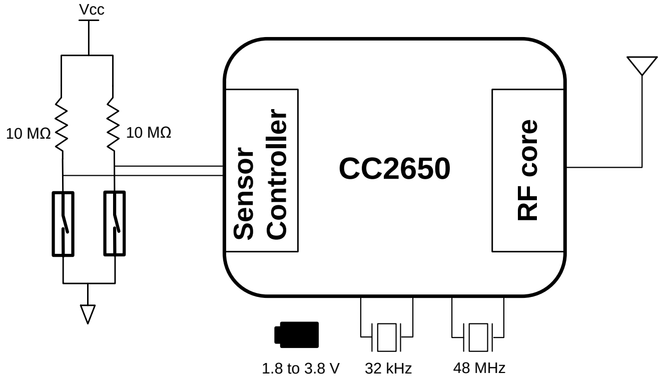
ASK Our E2E Experts
Bluetooth Low Energy Forum

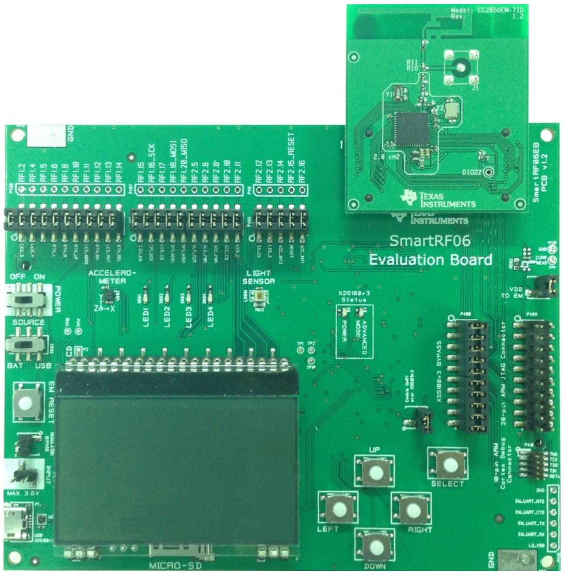
## Design Features

- <11-µA Standby Current for Both Bluetooth Connected and Advertising Mode
- Nine-Year System Operation With Button Cell Battery
- SimpleLink™ *Bluetooth*® CC2650 Wireless MCU Provides a Single-Chip Solution to Measure and Communicate
- Two Reed Switches to Detect Rotations

## Featured Applications

- Water Meters
- Gas Meters
- In-House Water Usage Monitor
- Automatic Meter Reading Within 10 Meters
- Rotation Detection with Counter and Odometer





An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

All trademarks are the property of their respective owners.

# 1 System Description

This TI Design is an elegant single-chip solution with minimal component count for mechanical water or gas meters. This system uses two reed switches to detect the rotating magnet inside the meter. The magnet is confined in the casing and drives the external gear counter through the magnetic flux. With the reed switches, the total gear counter can be replaced without the need to redesign the existing casing. *Bluetooth* low energy (BLE) communications provide users with the ability to read the meter through a smartphone or PC at any time within 10 meters. Automatic daily or hourly monitor of usage becomes possible.

Two reed switches can detect rotational direction. When opening a valve, the rotor in the meter will rotate in one direction, but it turns in the opposite direction when the valve is closed. This is due to the water bouncing backwards. The system will count up and down for these two directions to avoid double counting.

The CC2650 is a highly integrated processor with an RF core. This TI Design mainly uses the sensor controller, main CPU, and the RF core of the CC2650 (see Figure 1). The sensor controller engine, a module of the CC2650, handles interrupts from the reed switches and records the rotational number. The main CPU running with TI-RTOS is in sleep mode and only wakes up by a system timer or the RF core. The system timer, in a 5-second interval (configurable), triggers a timer event to wake up the main CPU, which then fetches the rotational count from the sensor controller and sends an notification through the RF core in connected mode. In advertising mode, the sensor controller works alone to count the rotation number while the RF core transmits an advertising signal.

The sensor controller consumes only 8.2 µA per MHz, lowering system power consumption and freeing the ARM® Cortex®-M3™ CPU to work in parallel to run the communications stack. System performance is improved as the sensor controller continues to measure the rotation of the rotor while the main CPU is triggered by the RF event.

The RF core has its own CPU for processing. The *Bluetooth* firmware stack, smartphone BLE app, and the BTool software operated in the PC are provided by Texas Instruments for easy system development.

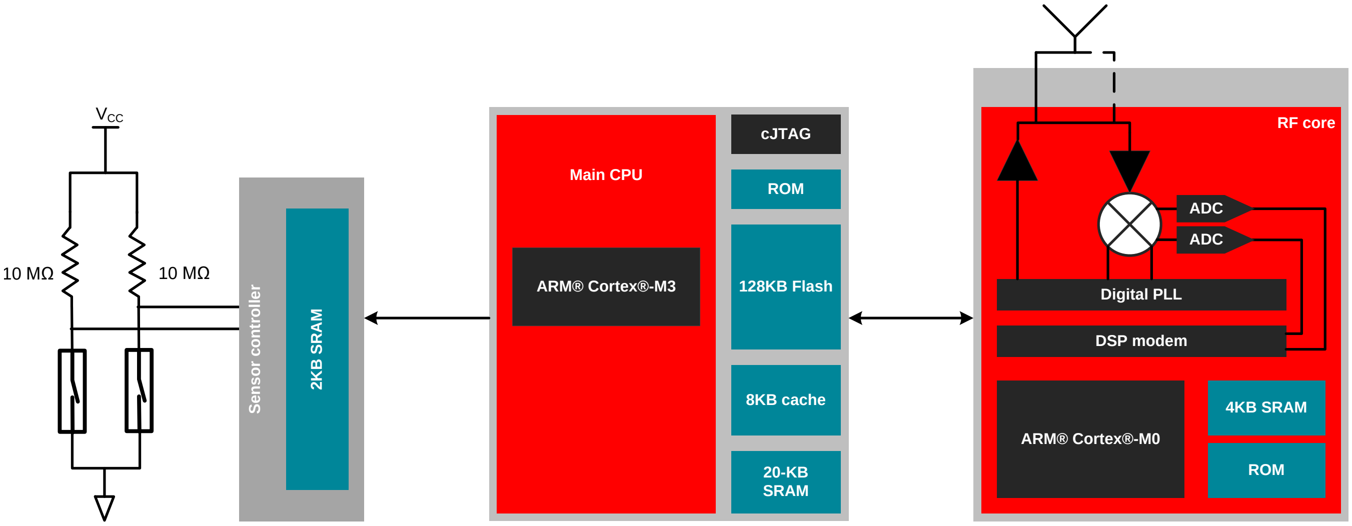


**Figure 1. Signal Flow Among Sensor Controller, Main CPU, and RF Core of CC2650**

## 2 Design Features

### 2.1 Low Power Design

The current consumption in standby mode is very low and mainly depends on the connection event of the RF core. The standby current of the CC2650 is 1 µA, not including the RF part. For two reed switches, the average is 0.2 µA. Figure 2 shows the profile of current consumption during a connection event.

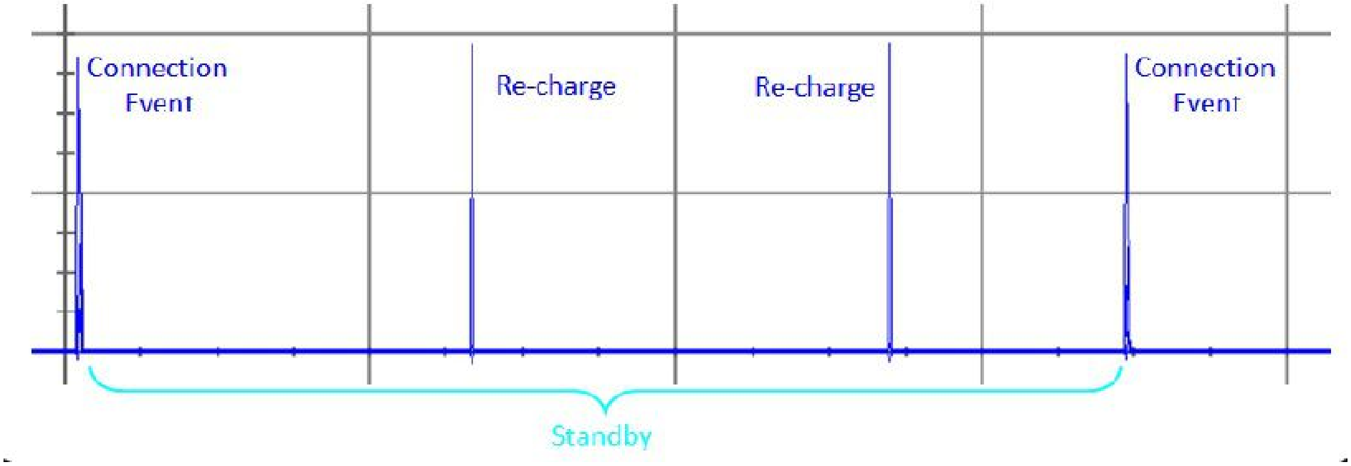


**Figure 2. Profile of Current Consumption During Connection Event**

When there is a connection event, for example when updating data, the RF signal is transmitting and receiving between the system and the RF reader like the smartphone or PC. The event lasts for less than 2 to 4 ms, and the peak current can be close to 10 mA (see Figure 2). For a paired system, the average current for one connection event in a period of one second is 8.78 µA. Together with the standby current, the total current consumption is 9.74 µA. See the CC2650 application note for more details (SWRA478).

To further reduce the current consumption, lengthen the time interval between each connection event. For example, for one connection event per 3 seconds, the average current for one event is less than 3 µA.

## 2.2 Single Chip Solution

The CC2650 is highly integrated with two ARM core CPUs, a RF core, a sensor controller, and various peripherals.

This TI Design uses the CC2650EM-71D evaluation board (see Figure 3). The only external components are two reed switches and two 10-MΩ resistors. The EVM has an onboard PCB antenna for BLE and two crystals, one of 24 MHz and the other of 32.768 kHz.

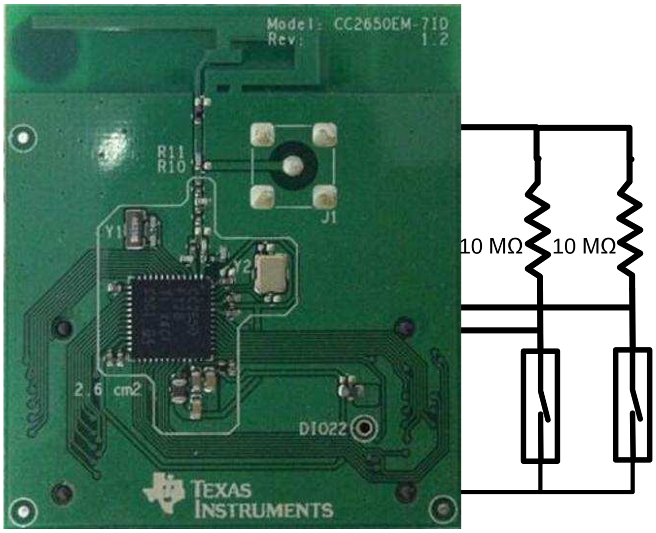


**Figure 3. Single Chip on CC2650EM-71D**

The CC2650 can operate with a voltage level from 1.8 to 3.8 V. Testing for this design used input voltages from 2.1 to 3.3 V with no impact on performance. A single button cell such as a standard CR2032 can power the board.

The SmartRF06 evaluation board is used as an extension board for the CC2650EM. It has a socket to connect to this board. The reed switches circuit is connected through the connectors on SmartRF06. In LCD mode of the firmware, data will be sent through SPI to the LCD module.

## 2.3 Bluetooth Low Energy Connection

An integrated RF core in the CC2650 is responsible for Bluetooth connection. This RF core has its own ARM CPU Cortex M0 and RAM sector for RF operation. TI provides the *Bluetooth* stack and API for this application.

## 2.4 Two Reed Switches for Rotational Detection

Two reed switches are used to detect rotational direction and avoid false counting due to the mechanical rotational vibration of magnetic rotor, which turns the reed switch on and off.

The reed switches are pulled up with 10 MΩ (see Figure 3). This can minimize the current consumption when the switch is closed. It also increases the RC constant with the reed switch being seen as a capacitor. With a high RC, the bouncing effect of the switch will not generate a detectable signal to make false trigger to the system.

Although this TI Design uses two sensors, the algorithm and operation of the reed switch with a sensor control engine in the CC2650 are different to the e-scanIF with two LC sensing detection in the TI Design TIDM-LC-WATERMTR. The operation of reed switches is as follows:

The reed switch is opened when facing with a single north or south pole, but closed when facing the boundary of the poles (see Figure 4).

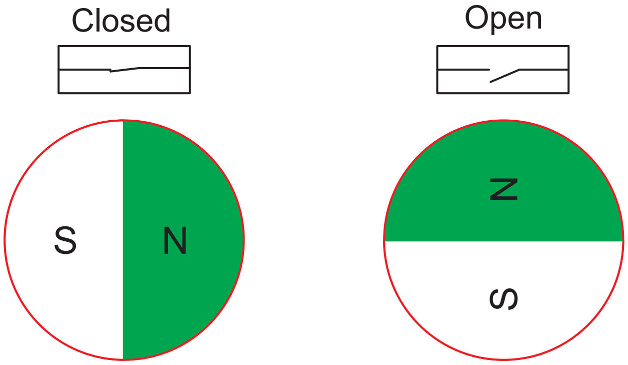


Figure 4. Reed Switch Over a Magnetic Rotor

To detect rotational direction, two reed switches have to be placed in 45 degree apart as Figure 5. The user has to check if the two signal output are in 90 degrees out of phase. Sensor 2 will be connected to the DIO1 of CC2650, and sensor 1 to DIO 23. These two I/Os will be mapped to the sensor controller, but it will only check the edge trigger from sensor 2. As there is a delay of around 400 µs from the trigger to the active mode of the sensor control, the location of the sensors are better adjusted that sensor 2 generates an edge signal 400 µs earlier before reaching 90 degrees out of phase.

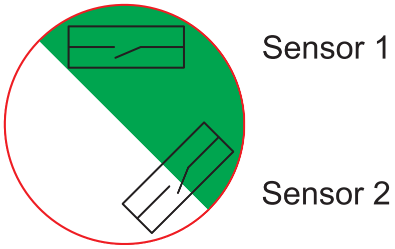


Figure 5. Two Reed Switches on Magnetic Rotor

Detecting rotational direction is based on the states of sensor 1 and sensor 2. Two signals with 90 degrees out of phase to each other provide four sequential states denoted in $S_1S_2$ as "00", "10", "11", and "01". This sequence repeats itself for a forward direction and goes in the opposite sequence when moving in backward direction. To detect the state change with high efficiency and minimal resources, the following algorithm is invented for this TI Design.

1. Only active when there is an edge change, falling or rising edge, of sensor 2 is detected.
2. For forward direction (see Figure 6), $S_1S_2$ follows the pattern of "10", "01", "10", "01", and so on.
3. For backward direction (see Figure 6), $S_1S_2$ follows the pattern of "00", "11", "00", "11", and so on.

Three look-up tables are constructed as follows:

- plusOneLUT[8] = { 0,0,1,0,0,1,0,0 }
- minusOneLUT[8] = { 1,0,0,0,0,0,0,1 }
- nxtEdgeTrigger[8] = { 1,0,1,0,1,0,1,0 }

The pointer to the look-up table is a three-bit address of [nxtEdgeTrigger, S1, S2].

The value of plusOneLUT is added into the count-up counter (upCounter), which records the number of counts in forward direction, while minusOneLUT is added into the countdown counter (downCounter), which records the number of counts in backward direction. The effective number of counter (effCounter) equals to upCounter – downCounter. For nxtEdgeTrigger, the "1" indicating the setting of next trigger will be rising edge, "0" for falling edge.

- upCounter += plusOneLUT[n]
- downCounter += minusOneLUT[n]
- effCounter = upCounter – downCounter

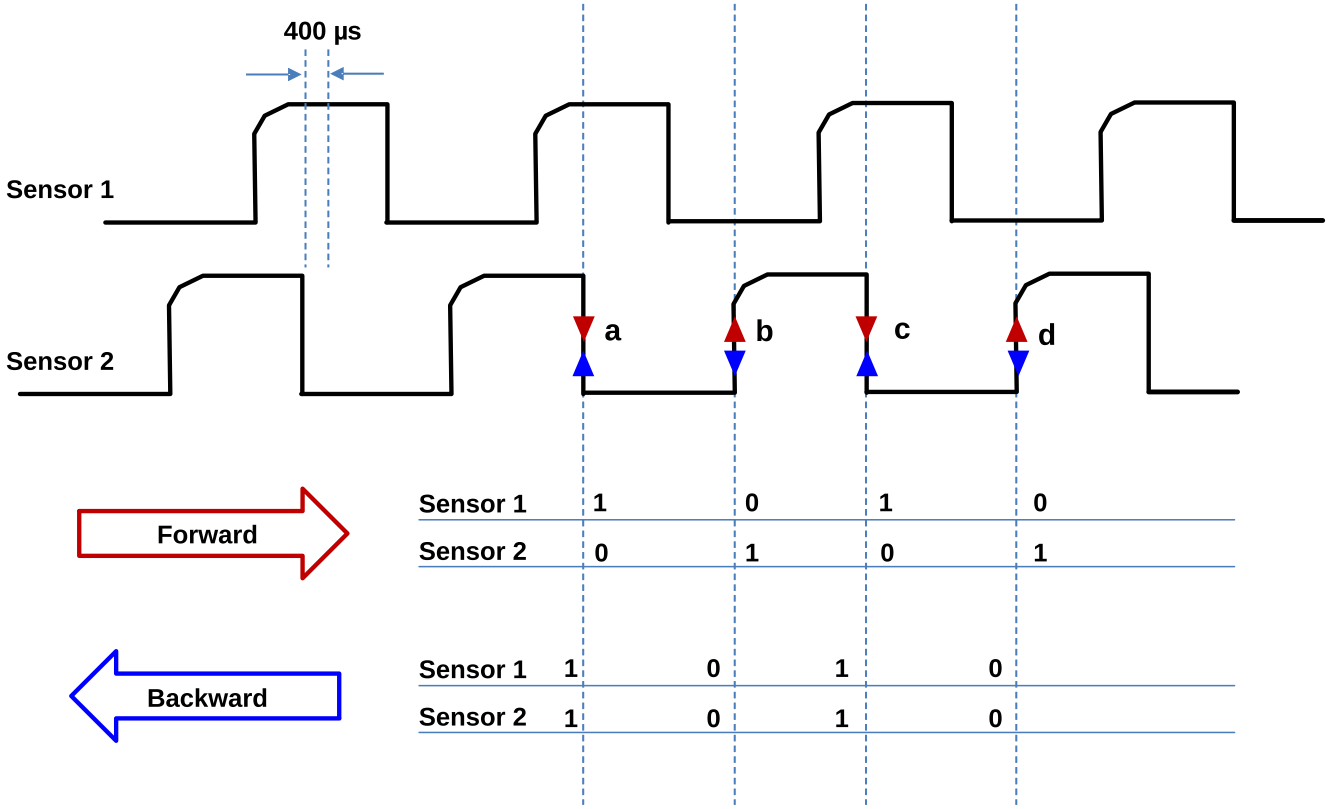


**Figure 6. Direction Detection With Two Sensors**

### 2.4.1 Working Example

In a water meter, a valve is opened. It starts with a falling edge "a" from sensor 2. When a falling edge at "a" is detected, sensor 1 will see "1" and sensor 2 will see "0". The address [0,1,0] is pointing to the number 2 in the table plusOneLUT[2], which has "1", meaning that upCounter will be added by one. Table minusOneLUT[2] has "0", meaning that downCounter will be added by zero. Table nxtEdgeTrigger[2] has "1", meaning that the next edge trigger is set to rising edge.

For rotating in forward direction, rising edge "b" will be detected. S1 gets "0", S2 gets "1". The table address of [1,0,1] is pointing to number 5 of the look-up tables. The upCounter will be added up by one, while the downCounter will be added up by zero. The next trigger will be set to falling edge. This will keep on counting in forward direction with detecting edges "a", "b", "c", "d", "a", "b", and so on. In this way for one complete rotation, upCounter will be added up by 4. downCounter will be always added up by zero and so there is no change for it.

When the valve is closed, the water flow is bounced backward (for example, at the point between "b" and "c"). At this point after edge of "b", the sensor controller is set to detect the next falling edge. The expected next edge will be "c" if it keeps on forward movement. However, now moving backward, the falling edge at "b" is detected instead. Therefore, both sensor 1 and sensor 2 get "0". The address "0,0,0" is pointing to the number zero of the tables. Here now, minusOneLUT[0] is "1", leading to an add up of downCounter; for upCounter, it adds a zero, as plusOneLUT[0] is zero. The resultant effCounter is in a result with a minus of one. The next edge trigger is rising edge. Keeping in backward direction, downCounter will keep increasing by one and upCounter does not change.

A special situation is on the rotational vibration of rotor, for example vibrating around the edge "b". In this way, when it moves forward, upCounter adds one. When it moves backward, downCounter adds one. In the result, there does not have a real displacement, which is reflected by the effCounter.

This algorithm is implemented in the sensor controller of the CC2650.

## *2.5 Low Standby Current for Both Connected or Disconnected Mode*

The *Bluetooth* board can connect to a smartphone or PC. When connected, a periodic connection event is established, which takes the largest portion of current consumption. To reduce this consumption, the time interval between each connection event can be adjusted to lower the total current to a single-digit microampere level. The interval setting is done in the smartphone or PC. This TI Design takes 10.8 µA with one connection event per 2 seconds.

When there is no RF connection, the Bluetooth device will keep transmitting periodic signals to broadcast its existence. This process is called advertising. Each advertising signal, lasts for a few milliseconds, takes the largest amount of total standby current. To reduce it to a low single-digit microampere level, use the same technique of lengthening the time interval between each advertising signal. This setting is done in the CC2650. In this TI Design, it takes 10.4 µA with one advertising signal in every 3.3 seconds.

## *2.6 Button Cell Battery Powered*

By calculation, this TI Design can keep running for nine years with four CR2032 cells. The calculation is based on a water meter with its rotor running at full speed of 50 rotations per second for one hour every day, and the rest of time is in standby mode with *Bluetooth* connected all the time. One hour of full speed rotation will measure the maximum flow rate of a meter. A typical mechanical water meter will output 1600 to 2500 liters per hour at its maximum rate. The self-discharge rate of the CR2032 is 1% per year.

## 3 Block Diagram

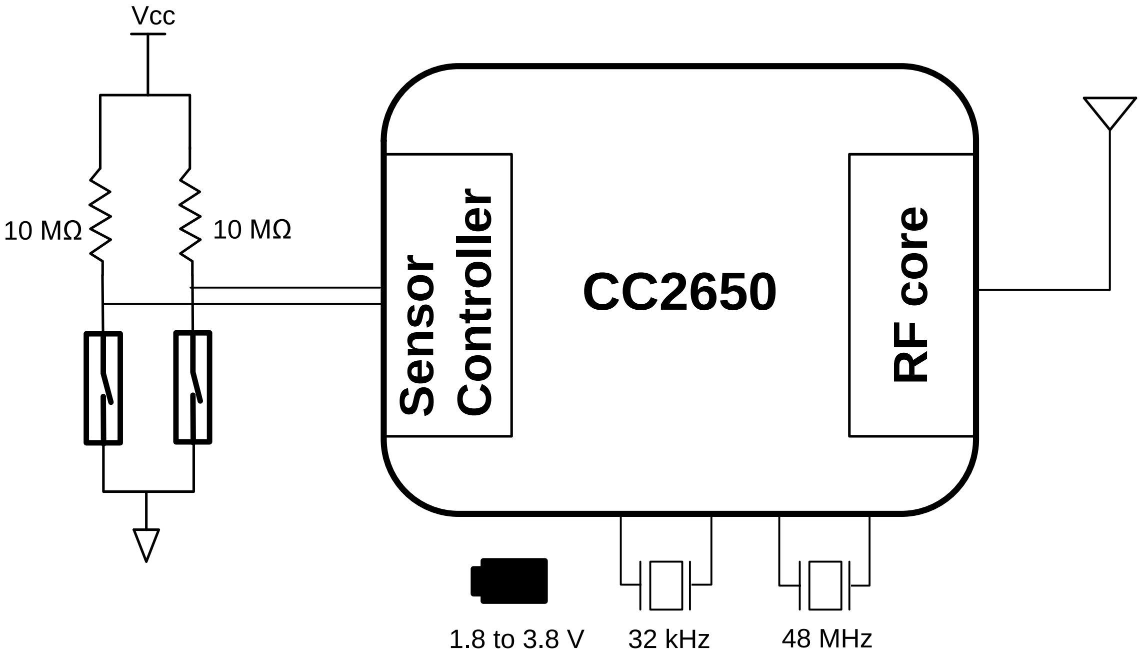


**Figure 7. System Block Diagram**

# 4 Circuit Design

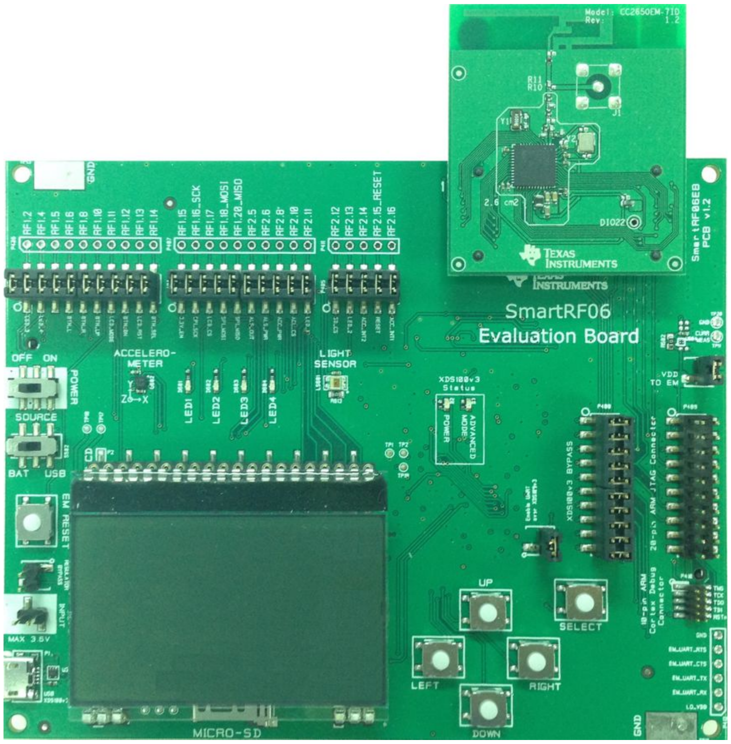


**Figure 8. CC2650EM-71D Plugged Onto SmartRF06 Evaluation Board**

This TI Design hardware is built with the evaluation boards of the CC2650. For easy connection and current measurement, the CC2650-71D (CC2650 evaluation board) is plugged into SmartRF06 evaluation board (see Figure 8). All external circuit is connected through the SmartRF06. The SmartRF06 has XDS100v3 debug probe with which the firmware will be loaded to CC2650. Two 10-MΩ resistors and two reed switches are connected externally with a breadboard (see Figure 9).

The voltage supply to the external circuit is from the connection pin, pin 7 or pin 9, of the CC2650-71D, connecting to the VDD_EB of the board. Two input signal wire is connecting to DIO_1 and DIO_23 of the board. For detail pin number of the connector on the evaluation board, see the schematic diagram of the SmartRF06 and CC2650EM-71D.

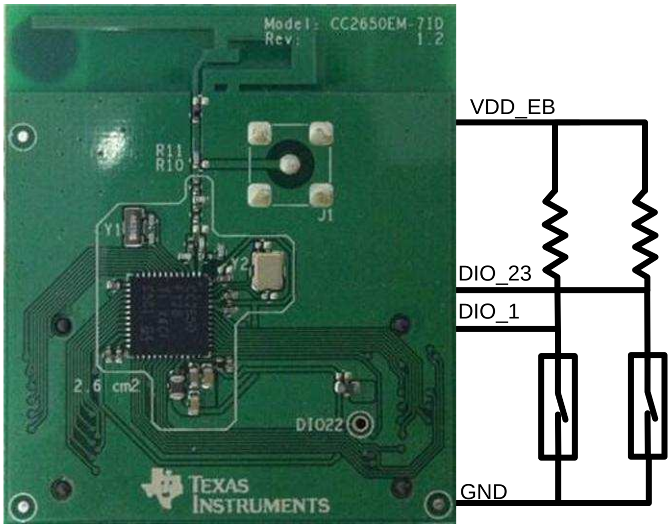


**Figure 9. Circuit Connection to CC2650EM-71D**

A high RC constant can eliminate the bouncing effect of the switch. Depending on the level of self-capacitance, connect a small capacitor of 100 pF in parallel to the switch if the self-capacitance is too small. However, in this TI Design, as the switch used has a high enough self-capacitance, it does not need an extra capacitor.

The wires connecting DIO_1 and DIO_23 have to be short and placed a good distance apart from each other to avoid signal interference.

## 5 Software Installation

The following firmware development tools and BLE Software package has to be downloaded and installed to start this project:

- Firmware package of this TI Design
- Code Composer Studio™ version V6.1 (CCS61)
- BLE software stack, BLE-STACK-2-1
- Sensor Controller Studio
- SmartRF Flash Programmer 2
- BTool
- Smartphone App, TI SensorTag (Android, iOS)

The firmware package can be downloaded from the TI Design folder TIDM-BLE-REEDMTR. Unzip it into "c:\ti\TIDM-BLE-REEDMTR". Three folders can be found: "Application", "Profiles", and "SCIF_CODE".

Install CCS61 in the folder "c:\ti" and select "SimpleLink Wireless MCUs" in Processor Support. After that, install the BLE software package into the default folder "c:\ti". From this software package, it will install three folders: "simplelink", "tirtos_simplelink_2_13_00_06", and "xdctools_3_31_01_33_core".

Install the software BTool in a different folder. Its setup file can be found inside the folder "simplelink" (C:\ti\simplelink\ble_cc26xx_2_01_00_44423\Accessories\BTool).

Install Sensor Controller Studio and SmartRF Flash Programmer 2 in different folders.

The firmware of this TI Design project has two parts. Both are modified from the example code of "SimpleBLE Peripheral".

Before starting the firmware modification, it is very important to backup the folder "SimpleBLE Peripheral", found in the path "C:\ti\simplelink\ble_cc26xx_2_01_00_44423\Projects\ble". Also backup the folder "SimpleProfile", found in the path "C:\ti\simplelink\ble_cc26xx_2_01_00_44423\Projects\ble\Profiles".

To import the example, open CCS61 and click "View" in menu bar to select "Resource Explorer (Examples)". In the "TI Resource Explorer" tab, click "BLE-Stack for SimpleLink wireless MCUs", select "CC2640F128", then "Projects", and then select "Simple BLE Peripheral". There are two CCS projects under this folder, "SimpleBLEPeripheralApp" and "SimpleBLEPeripheralStack". Import them one by one into the CCS workspace by clicking "Step1: Import the example project into CCS", shown in the window next to the folder list under the same tab.

There are two projects imported into CCS. Right-click onto the project and rename "SimpleBLEPeripheralStack" to "RS_FlowmeasureStack" and "SimpleBLEPeripheral" to "RS_Flowmeasure".

After these tools and software have been installed, keep the project "RS_FlowmeasureStack" unchanged. The following procedures are to delete some of the original source code in project "RS_Flowmeasure" and replace with the ones in the firmware package at "c:\ti\TIDM-BLE-REEDMTR". During the modification procedure, to add-in a file, click on the selected folder of the project, then click on "Project" in menu bar and select "Add files". Make sure to add in with a link; do not copy into the folder.

Delete and add in the following links in project "RS_Flowmeasure”:

1. In the Startup folder, delete the link to "main.c". Copy "main.c" from the Application folder of the firmware package to "C:\ti\simplelink\ble_cc26xx_2_01_00_44423\Projects\ble\SimpleBLEPeripheral\CC26xx\Source\Application\", and then add in these files to the folder.
2. In the Application folder, delete the link to "simpleBLEPeripheral.c" and "simpleBLEPeripheral.h". Copy the following files from the Application folder of the firmware package to "C:\ti\simplelink\ble_cc26xx_2_01_00_44423\Projects\ble\SimpleBLEPeripheral\CC26xx\Source\Application\", and then add-in these files to the folder:
   - RS_Flowmeasure.c
   - RS_Flowmeasure.h
   - scif_framework.c
   - scif_framework.h
   - scif_main_tirtos.c
   - scif_main_tirtos.h
   - scif_osal_tirtos.c
   - scif_osal_tirtos.h
   - scif.c
   - scif.h
3. In the Profiles folder, delete "simpleGATTprofile.c" and "simpleGATTprofile.h". Then copy "flowGATTprofile.c" from the Profiles folder of the firmware package to "C:\ti\simplelink\ble_cc26xx_2_01_00_44423\Projects\ble\Profiles\SimpleProfile\CC26xx\". Add in this file to the Profiles folder of the project.
4. Copy "flowGATTprofile.h" from the Profiles folder of the firmware package to "C:\ti\simplelink\ble_cc26xx_2_01_00_44423\Projects\ble\Profiles\SimpleProfile\". Add in this file to the Profiles folder of the project.

To compile the code, click on the project title "RS_Flowmeasure", right click and select "Rebuild Project" or "Build Project". After completing the compilation, click on the project title "RS_FlowmeasureStack" and right click to select "Build Project".

After that, prepare the SmartRF06 Evaluation Board with the small board CC2650EM-71D plugged onto it. Connect the SmartRF06 board to the notebook and turn the slide switches "Power" to ON and "Source" to USB. Now, download the project "RS_FlowmeasureStack" into the board by selecting “Run” in the menu bar and select "Debug". It does not run as this is part of the complete software. Stop it and go back to CCS edit mode. Download the project "RS_Flowmeasure" into the board in the same way. Now, it will run.

To test if the firmware works properly, download the smartphone app TI SensorTag. Open it with *Bluetooth* switched on. On the smartphone screen, select "RS Flow Measure", which is under "Bluetooth Smart Devices". Then choose "Service Explorer". Select "1" and then "1" again. Set notify state for characteristic to ON. The value will show the number of rotation in Hex format, with the low byte come first. For example, <0518> should be read as 0x1805, which is 6149 in decimal format.

# 6 Firmware Description

The project "RS_FlowmeasureStack" contains all Bluetooth stack and all files inside it will not be changed. Only the files in project "RS_Flowmeasure" are modified.

In the Application folder of project "RS_Flowmeasure", all the files with prefix of "scif" are generated from Sensor Controller Studio. The source code and project setup files for Sensor Controller Studio can be found in "SCIF_CODE" of firmware package.

The scif-code is running in the sensor controller engine in the CC2650 without the convention of the main CPU. Its function is to detect the interrupt from the sensor and count the number of rotations as described in Section 2.4.

To check or modify the code, open existing project with Sensor Controller Studio and select "Reed_Switch.scp" in the SCIF_CODE folder. All the setup and code will be imported into Sensor Controller Studio. User guides can be found in the "Tool Documentation" in the "Start Page" of the studio.

This code module uses two digital input pins and GPIO Event Trigger. The code is divided into four parts, "Execution Code" and "Termination Code" are not used. In the Initialization code, this code only runs for once time only at the start up. It builds the tables of plusOneLUT, minusOneLUT and nxtEdgeTrigger. For details, see Section 2.4.

When there is an interrupt trigger from sensor 2, the "Event Handler Code" will be called. The variable "state.upCounter" will count the forward rotation and will count up with 4 per complete rotation. "state.downCounter" will count the backward rotation. The resultant number of rotation is stored in "state.effCounter". "output.upCounter", "output.downCounter", and "output.effCounter" are the mirror of the corresponding variables in state.xxxxxx.

"Code Generator" in the menu column of Sensor Controller Studio generates the source code by clicking the "Generate driver source code" button. The user can then move the six generated files (scif.c, scif.h, scif_framework.c, scif_framework.h, scif_osal_tirtos.c, and scif_osal_tirtos.h) into the Application folder of the firmware package (do not change the scif_main_tirtos.c/.h files) and follow Step 2 in Section 5 to integrate the code into the BLE application folder.

To get the output values from the sensor controller, use the function "RS_Flow_SCIF_performPeriodicTask()" in scif_main_tirtos.c . This function also calls another function to store the data from effCoutner into the value of "Characteristic 1" of the BLE service.

To understand the current consumption of the sensor controller, go to the E2E forum topic "The current consumption for the sensor controller of cc2650".

## 6.1 Firmware in RS_Flowmeasure

The "main.c" is located in the Startup folder. After initialization, the system starts the BIOS and begins the advertising mode of the BLE. The time interval of the advertising signal is configurable by setting the constant of "DEFAULT_ADVERTISING_INTERVAL" in "RS_Flowmeasure.c". The default value is 3.3 seconds.

Besides the advertising signal, the only working module is the sensor controller engine. After connecting to the smartphone or PC, a timer starts and triggers a periodic event in the time interval set by the constant "RSF_PERIODIC_EVT_PERIOD" in "RS_Flowmeasure.c". The default value is 5 seconds.

Responding to this timer event is the task "RS_FlowBLEPeripheral_taskFxn()", which will then call "RS_Flow_SCIF_performPeriodicTask()" to get the data from the sensor controller, and then call "RS_FlowBLEPeripheral_performPeriodicTask()" to send out notification with an integer value in characteristic 2.

For the detailed modification procedure from "simpleBLEPeripheral" to "RS_Flowmeasure", see a similar modification example in "Tutorial: How to Create a Custom Bluetooth Smart Embedded Application with the CC2650DK". The main difference in this TI Design is using the name "RS_Flow" as the prefix instead of sunlight in the example.

# 7 Test Setup

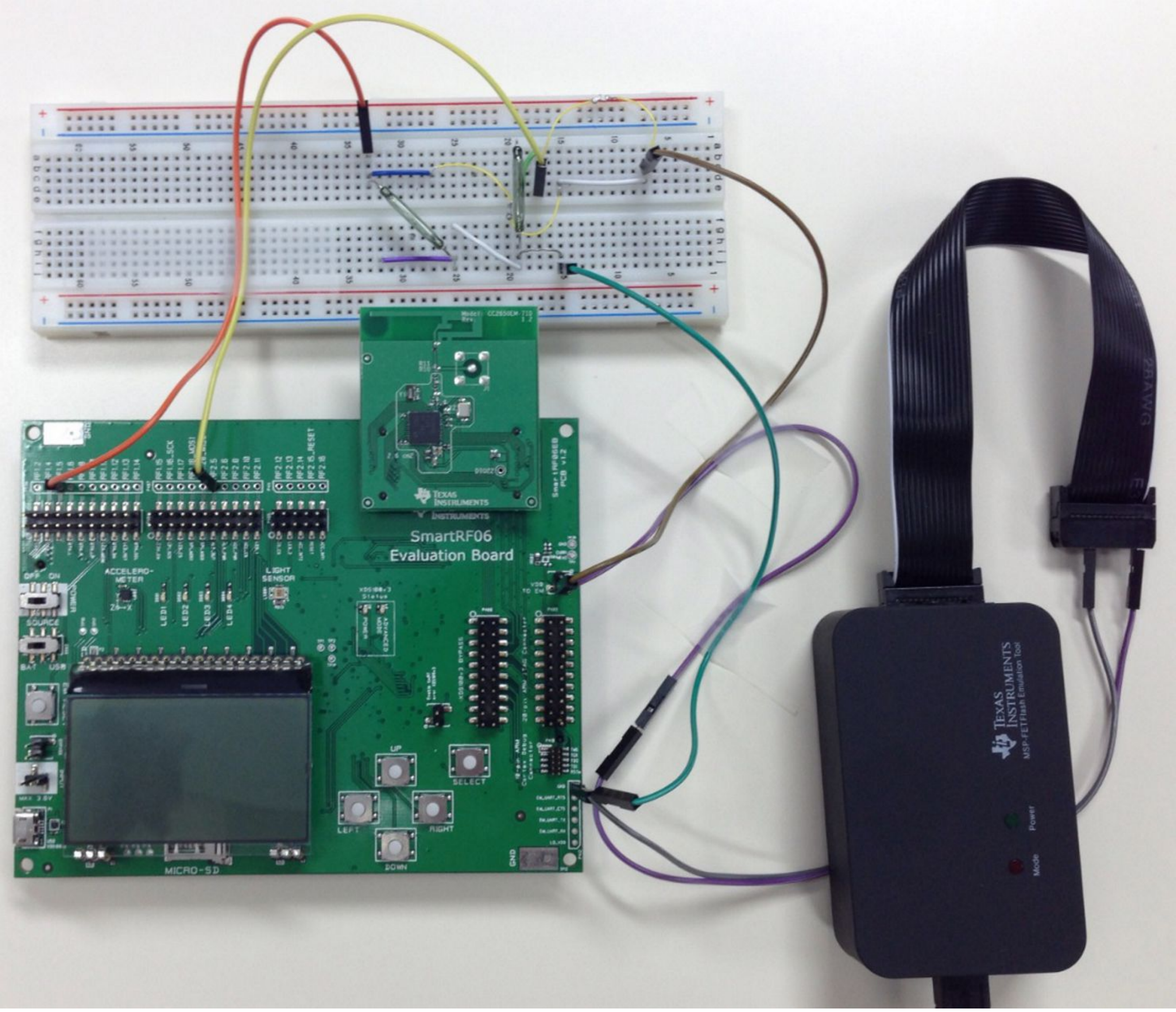


**Figure 10. Test Setup With EnergyTrace**

### 7.1 Test Setup to Measure Current Consumption

The reed switch circuit is constructed on a breadboard (see Figure 10) and connected to the SmartRF06 evaluation board to measure the system current consumption. Procedures are as follows:

1. Follow Section 5 for all software installation, configuration, and loading the firmware of RS_FlowmeasureStack and RS_Flowmeasure into the CC2650.
2. In CCS61, in the project properties, go to Build → ARM Compiler → Advanced Options → Predefined Symbols to remove "TI_DRIVERS_LCD_INCLUDED".
3. In project RS_Flowmeasure, go to Application → scif_main_tirtos.h, and change the setting to "#define LCD_ON 0".
4. Rebuild the project and then load the firmware into the CC2650.
5. Disconnect the USB cable from the computer.
6. Remove all jumpers from the SmartRF06 evaluation board.
7. With the orange wire, connect sensor 2 to RF1.5 of the SmartRF06 board (see Figure 10).
8. With the yellow wire, connect sensor 1 to RF2.5 of the SamrtRF06 board.
9. With the green wire, connect the ground of the board to the reed switch circuit.
10. With the brown wire, connect the $V_{DD}$, at the pin "VDD TO EM", of the board, to the reed switch circuit.
11. Connect the $V_{CC}$ from the MSP-FET Tool with built-in "EnergyTrace" to the $V_{DD}$ of the board and the ground of the MSP-FET to the ground of the board.
12. Use the EnergyTrace controller with CCS61 to test the current consumption in advertising mode, and in connection mode with a different rotational speed.

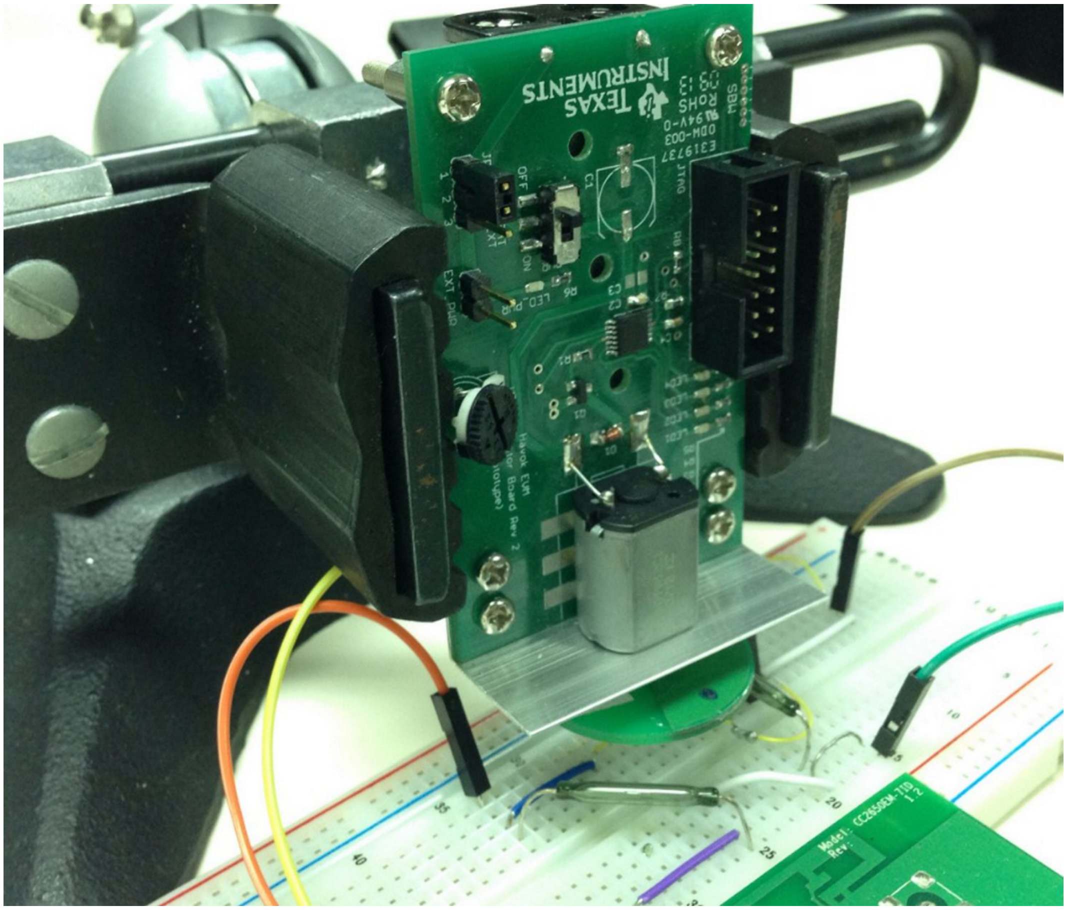


**Figure 11. Motor Board**

In the test setup, a DC motor board emulates the rotor rotation in a flow meter. There is a magnet attached onto the rotor plate. Its rotational speed can be adjusted.

This setup can measure the current consumption of the system in advertising mode, with rotational speed from 0 to 100 rotations per second, by varying the motor speed.

To measure the current consumption in *Bluetooth* connected mode, use the BTool to adjust the time interval between connection events. Find a detailed technical explanation in the application report *Measuring Bluetooth® Smart Power Consumption* (SWRA478).

The BTool setup is as follows:

1. Use another SmartRF06 board with CC2650EM-71D connected.
2. Use "SmartRF Flash Programer 2" to program the board with the Hex code "CC2640_SmartRF-71D_HostTestRelease_All.hex" located at "C:\ti\simplelink\ble_cc26xx_2_01_00_44423\Accessories\HexFiles".
3. Open the software "BTool" with default setup, which is a baud rate of 115200, "none" in Flow and Parity, 1 stop bit and 8 data bits.
4. In the "Discover / Connect" tab in the BTool window, set the "Min and Max Connection Interval" to 2000 ms, and click "Set".
5. Click "Scan" to discover the BLE device.
6. When the slave address found, click "Establish". Connection info will be shown in the left most window of BTool when connected.
7. Move the mouse cursor onto the "Handle" under "connection info", right click and select "Discover UUID" or "Read Values". The UUID and values of the reed switch BLE setup will be fetched and display onto the BTool. If there is a time out message prompt out, click "OK", then click "No" to the following warning message.
8. In the bottom lines of fetched data, right click on the "value" of "Client Characteristic Configuration" or double click on it to select "Write Item". A window is prompt out. Type in 01:00 in the "Value" box and click "Write Value". The system will notify the BTool the updated rotational number per every 5 seconds (default, but configurable with RSF_PERIODIC_EVT_PERIOD in RS_Flowmeasure.c).

## 8 Test Results

The setting of the *Bluetooth* device to detect the reed switch with the sensor control module of the CC2650 is as follows:

- In advertising mode, it is set to post an advertising signal by each 3.3 seconds.
- In connection mode with BTool, it is set to take 1 connection event per 2 seconds and 1 notification in every 5 seconds.

Using the EnergyTrace, the current consumption is measured as follows: For each point in the graph (Figure 12) it is a mean value of measurement in a period of 10 seconds.

- In advertising mode, 10.3 µA with no rotation
- In connection mode, 10.8 µA with no rotation

When testing with rotating rotor, the current consumption depends on the rotational speed. It takes 0.593 µA for each rotation.

For a typical water meter, the maximum rotational speed is 50 rotation per second (rps). In connection mode, it takes a total of 10.8 + 0.593 × 50 = 40.45 µA When there is no flow, it takes 10.8 µA.

When the connection event appears once every 60 seconds, It will only take 2 µA for no flow and 31.65 µA for 50 rps.

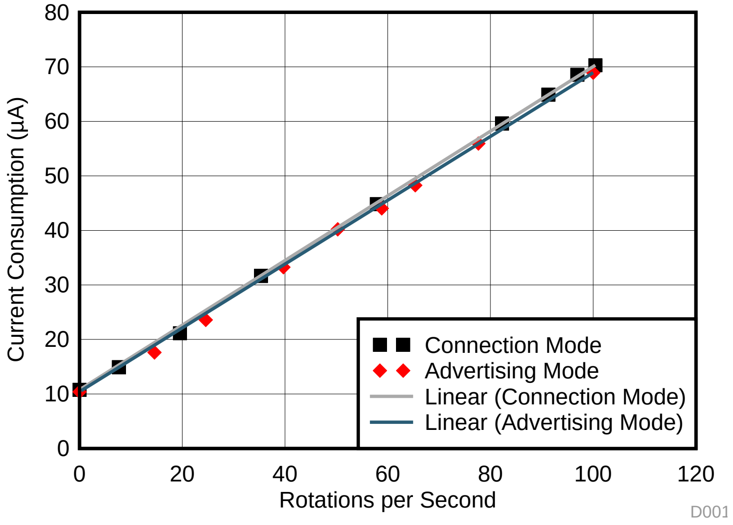


**Figure 12. Test Results of Advertising Mode and Connected Mode**

Table 1. Test Results of Advertising Mode and Connected Mode

| RPS | CONNECTION MODE (µA) | ADVERTISING MODE (µA) |
|---|---|---|
| 0.0 | 10.8 | 10.4 |
| 7.7 | 14.9 | |
| 14.6 | | 17.6 |
| 19.5 | 21.1 | |
| 24.6 | | 23.6 |
| 35.3 | 31.6 | |
| 39.7 | | 33.3 |
| 50.3 | | 40.2 |
| 57.9 | 44.8 | |
| 58.8 | | 44.0 |
| 65.4 | | 48.3 |
| 77.6 | | 56.0 |
| 82.2 | 59.7 | |
| 91.2 | 64.9 | |
| 96.9 | 68.6 | |
| 100.0 | | 68.9 |
| 100.4 | 70.3 | |

Based on the test results of the system current consumption, standby mode in 10.8 µA and rotating mode in 0.593 µA per each rotation, it can formulate a table of battery life time in number of years in Table 2, varied with the number of CR2032 battery cells and the average number of operating hours running at full speed of 50 rps.

Note that the overall current consumption is dominated by the standby current. The user can lengthen the battery lifetime by lowering the standby current.

Table 2. Battery Life Time in Years

| NUMBER OF CELLS CR2032 | NUMBER OF HOURS RUNNING IN FULL ROTATION SPEED IN EACH DAY | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.5 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 2.4 | 2.2 | 2.0 | 1.9 | 1.7 | 1.6 | 1.5 | 1.4 | 1.3 | 1.2 | 1.2 |
| 2 | 4.7 | 4.4 | 4.0 | 3.7 | 3.4 | 3.2 | 3.0 | 2.8 | 2.6 | 2.5 | 2.3 |
| 3 | 6.9 | 6.6 | 6.0 | 5.5 | 5.1 | 4.7 | 4.4 | 4.1 | 3.9 | 3.7 | 3.5 |
| 4 | 9.2 | 8.7 | 7.9 | 7.3 | 6.7 | 6.3 | 5.8 | 5.5 | 5.2 | 4.9 | 4.6 |

## 9 Software Files

To download the software files, see the design files at TIDM-BLE-REEDMTR.

## 10 About the Author

**THOMAS KOT** is a System and Application Engineer at Texas Instruments. He works on the flow meter reference design development and customer support. He has published various flow measurement TI Design and application reports. Thomas received his bachelor of engineering in electronic engineering from Hong Kong Polytechnic University in 1995 and received his master of science in electronic and information engineering from the same university in 2005. He received his master of business administration from City University of Hong Kong in 2007. Contact him at thomas-kot@ti.com.

# IMPORTANT NOTICE FOR TI REFERENCE DESIGNS

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265