Date:	20 July 2021
Revision:	0.0.5
Author:	David Witten
Pages:	14

Configuring the Raspberry Pi 3 or 4 for use with the runMag utility and the TangerineSDR ‘Pear’ boards

Contents:

Introduction

This document concerns the software setup for the Magnetometer Support Boards (MSB's) designed as prototypes to test the PNI rm3100 magnetometer modules and the 'Pear' boards created for the TangerineSDR project. The primary software described here is a utility called 'runMag' that is available as source code. It is covered by the GPL 3 open source license, It may be downloaded from GitHub at https://github.com/wittend/rm3100-runMag.

It must be emphasized that while this software has so far proven useful and reliable, it was intended for testing and evaluation of these boards. ***Its author does not consider it production-grade code.*** Comments and suggestions are appreciated. But new work is focused on other tools and it is expected that this code will be superseded.

The 'runMag' utility was written for portability and does not invoke Raspberry Pi specific libraries.

Setting I2C up on Raspberry Pi 3 or 4 single board computer (SBC) is usually fairly simple. Use with other similar boards is possible (Odroid N2(+), Nvidia Nano, etc.), that will not be addressed here.

Please, please try to stick to this scenario. What seem like small deviations lead to problems that are difficult to diagnose, both now and later. Different OS versions, personal software preferences, unique naming strategies all lead to an exponentially expanding support problem.

Installation of all software required to monitor, log, and archive data from either the Magnetometer Support Board prototypes or the HamSCI 'Pear' boards will be the focus of this document.

The PNI RM3100 Magnetometer module used here is NOT 5.5v tolerant on the I2C bus.

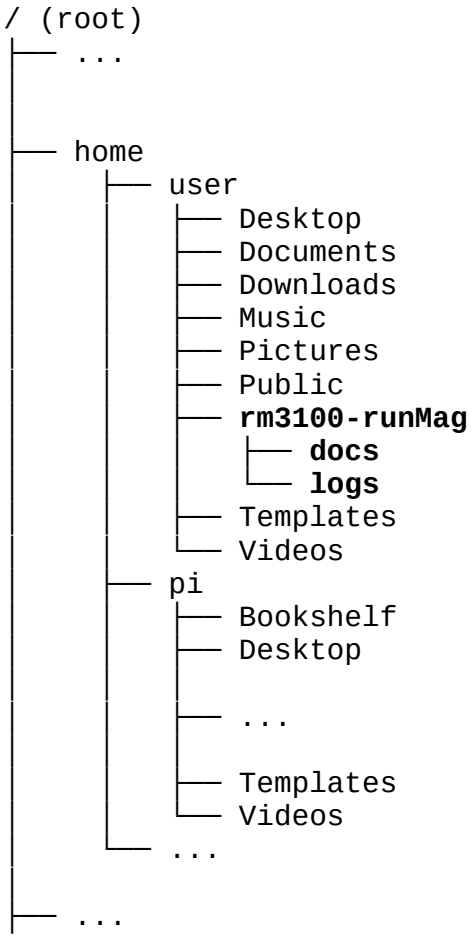
Warning: Applying 5v (or higher) to any other pins will damage the magnetometer.

This document also assumes that you have the following incidental items:

- For remote use - 100 foot shielded twisted pair CAT-6a cable: https://www.amazon.com/gp/product/B00HEM653S/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1

- Other items required for in-ground installation of the remote sensor. (See other documents detailing proper installation).

For these instructions to work, it is important to use this layout:



Setting I2C up to work on Raspberry Pi 3 or 4

For the purposes of this document I will assume that use the current desktop version of Raspbian OS (32 bit, armhf). The Raspberry Pi uses the same image both model 3B and model 4B.

We suggest:

> Raspberry Pi OS current with desktop:
>
> https://www.raspberrypi.org/software/operating-systems/
>
> Current version:
>
> Release date: May 7th 2021
>
> Kernel version: 5.10
>
> Size: 1,180

The other versions may work as well, but contain unwanted software or require different installation steps.

Please see installation instructions at:

Installing operating system images

Now you need to plug in monitor, keyboard, and mouse. I prefer a wireless keyboard/mouse combo. If possible, connect a wired network connection at first, but a reliable WiFi environment should work.

Then power up the Pi and let it guide you through the setup.

Then restart the Pi.

In recent versions, you can select Menu > Preferences > Raspberry Pi Configuration and then you get an applet thing. If you select the tab "Interfaces" you can turn on I2C and also select SSH. These are important to the proper operation of your sensor.

Alternatively, from a terminal window/command prompt, you can type

```
$ sudo raspi-config
```

and make these same changes.

Again, you should reboot the Pi.

Next, you should make sure that your Raspberry Pi is connected to the network. Then you need to open a terminal window and execute the command:

```
$ sudo apt-get update
```

Once this has completed without errors, you need to execute:

```
$ sudo apt-get upgrade
```

This command may take a while and generate quite a bit of activity, but it assures that your software is up to date and that current security patches are installed.

Yet again, you should reboot the Pi.

Change the default Password!

From a terminal prompt, you can type:

```
$ passwd
```

It is very important that you change the default password for the ‘pi’ account.

You should create an account other than the default pi account for your use when operating your device. It should have a strong password that you can remember. The command for this is:

```
$ adduser [USER ID]
```

NOTE: Only the ‘password’ and ‘repeat password’ fields actually matter. You can hit enter for the others, but enter ‘y’ to accept.

Your working account should have the same group privileges as the pi account, especially the ‘adm’, ‘sudo’, ‘ssh’, ‘dialout’, ‘i2c’, ’input’, ’netdev and ‘plugdev’ memberships. While still in the ‘Pi’ account you should add these memberships using:

```
$ sudo usermod -a -G adm,sudo,ssh,dialout,i2c,plugdev,input,netdev [USER ID]
```

I prefer to disable the pi account altogether.

Yet again, you should reboot the Pi.

Testing I2C

Assuming that you have completed the appropriate configuration above, the next step is to make sure the appropriate devices are now available.

Open a terminal window and type:

```
$ ls /dev/i2c*'
```

The Raspberry Pi boards should show: "/dev/i2c-1".

Here we use the command ‘i2cdetect’. If you enter this command and you get an error, you may have to install the package ‘i2c-tools’ using the command:

```
$ sudo apt-get install i2c-tools
```

Now you want to type the command

```
$ i2cdetect -y 1
```

You should see something like this:

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:         -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

The dashes are unoccupied addresses (hexadecimal) on I2C bus 1. As a general rule, on a newly configured Raspberry Pi, all addresses on bus I2c-1 will be empty.

The numbers or dashes indicate the presence (or absence) of devices at those addresses, for example at 0x37 (in C syntax)

Now shut down the Raspberry Pi and connect the local end magnetometer extension board (with the 40 pin Raspberry Pi extension header) onto the Pi expansion bus.

Local board with I2C Extender

Initially you may see only one device. It is on the local board. The device should be at 0x19 and is the MCP9808 precision temperature sensor on the local module.

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- 19 -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

When the shielded twisted pair Cat 6a cable is attached and 5v is applied to the local module the sensors on the remote board should appear as well. The one at address 0x18 is the remote MCP9808 precision temperature sensor. The one at address 0x20 is the remote magnetometer module.

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- 18 19 -- -- -- -- -- --
20: 20 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

It is the job of the software to request register values from these addresses and to format them into something we can read.

Note:

The TangerineSDR ‘Pear’ boards default to the the non-standard address address for the magnetometer module: ‘0x23’. A properly working pair of ‘Pear’ boards appears like this:

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- 18 19 -- -- -- -- -- --
20: -- -- -- 23 -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Note:

The ‘Pear’ boards require the use of the additional command line parameter ‘-M 23’ to read this address.

Note:

If you assemble your complete magnetometer kit with both boards but only see one address responding, this means that something about the remote board, PNI module, or Cat 6a cable is not correct. See the troubleshooting notes at the end, if necessary, contact me (wittend@wwrinc.com).

Note:

If you get NO response using i2cdetect, check the attachment of the LOCAL board to the Raspberry Pi. Make certain that both the board orientation and the pin alignment in the 40-pin connector are correct.

Install the runMag Utility

To install the actual data collection utility, follow these steps:

1. Make sure that you are logged into the Raspberry Pi under the account that will actually run the data collection.
2. Be certain that you are in your home directory by entering:

   ```
   $ cd ~/
   ```

3. Enter the command:

   $ git clone https://github.com/wittend/rm3100-runMag.git

4. Now enter:

   ```
   $ cd  rm3100-runMag/
   ```

5. now enter the command:

   ```
   $ make
   ```

6. If there are no errors (there should not be) enter:

   ```
   $ chmod a+x  runMag
   ```

7. Then you can test the result by entering:

   ```
   $ ./runMag -h
   ```

If all has gone well you should now see the list of command line options that runMag understands. Most of these were only used for testing, and of no concern to most users.

Note: The install process should have created a sub-directory called ‘logs/’. This is where the captured log data will go.

To see if your runmag setup works, enter the following command:

```
$ ./runMag -b 1 -j -A 10 -c 400 -U 140 -Z
```

Or (if you are using the ‘Pear’ boards)**:**

```
$ ./runMag -b 1 -M 23 -j -A 10 -c 400 -U 140 -Z
```

Setup the Linux ‘rsync’ Tools

The ‘rsync’ is normally present in Raspberry Pi OS. It allows efficient transfer of files using secure keys.

This is the procedure that we are using (so far) to regularly backup the output logs to an accessible place on the cloud server.

- Please contact me by email (wittend@wwrinc.com) and I will create an account.
- Once the account is set up, all that is needed is for you to log in once at 45.33.26.82 (via ssh) to **change your password**.
- Then you log out of the cloud server and follow these instructions from your Pi 3/4.

The process to set up rsync on your Raspberry Pi so that it does not require manual entry of a password is:

```
$ ssh-keygen
```

Note: When this asks you to enter the passphrase, just press the **enter** key. **Do not give any password here!**
Now enter:

```
$ ssh-copy-id -i /home/<local user id>/.ssh/id_rsa.pub <remote user
id>@45.33.26.82
```

Note: The above will ask the password for your account on the **remote** host. It will copy the public key automatically to the appropriate location. You can test it by entering:

```
$ ssh <remote user id>@45.33.26.82
```

Then I put this in a script file (`rsynscript`) and run it:

```
$ rsync -avz /home/dave/rm3100-runMag/logs/dave@45.33.26.82:/home/dave/data/KD0EAG
```

This is essentially:

```
$ rsync -avz /home/<pi user id>/rm3100-runMag/logs/
<remote user id>@45.33.26.82:/home/<remote user id>/data/<station id>
```

Replace the **<pi user id>** placeholder with your user id on the Pi.
Replace the **<remote user id>** placeholders with your user id on the on the cloud server.
Replace the **<station id>** placeholder with your prefix you are using for your station This may be a call sign, ‘Grape’ Station ID, or some other identifier unique to you and less than 18 characters.
I put this script file in my 'rm3100-runMag/' folder and make it executable using chmod a+x.
You can test it from the command line by entering:

```
$ ./rsynscript
```

Once this works reliably, we can look at making it run automatically with the system scheduler ‘cron’.

Set Up the ‘cron’ Facility

Virtually all Unix-like operating systems provide a facility for executing tasks at scheduled times. On my system the most familiar of these,’cron’, was not installed by default. Sometimes it is.

While systemd has all the functionality needed to schedule task execution, it is less familiar and more cumbersome than the older ‘cron’ system. Therefore, ‘cron’ and its associated ‘crontab’ command is implemented on top of the systemd tools, and may need to be installed using:

```
$ sudo apt-get install cron
```

```
$ sudo systemctl enable --now cron
```

‘cron’ historically writes messages to the user’s email, but we will including some redirection in these commands to echo errors to a file.

Now create a directory in your home directory called cronlogs to receive the results from the rsync jobs.

```
$ mkdir cronlogs
```

To set up scheduled tasks you enter (don’t use sudo):

```
$ crontab -e
```

The first time that the ‘crontab’ command is invoked in an account it requests that you select a text editor to use. Nano is a common choice for Raspberry Pi users. Note that ‘crontab’ files generally require fully qualified path names.

You are then presented with the current user’s personal ‘crontab’ file. All lines in the file are initially comments. In the editor you should add (at the end) the line:

10 */2 * * * /home/**[USER ID]**/rm3100-runMag/rsync.sh >> /home/**[USER ID]/**cronlogs/cronlog.txt 2>&1

Be sure to replace both occurrences of **[USER ID]** with YOUR login id.

This entry will run the rsync.sh script with your credentials and synchronize the log files to the cloud server every couple of hours. Now exit with save.

Next enter:

```
$ sudo crontab -l
```

(This should list jobs in the queue).

Setting up ‘start-on-reboot’ for runMag

If you want to set up your Pi to auto start the logging process after each reboot, follow these steps:

```
$ cronlog -e
```

To edit your user crontab file and add the equivalent of these lines:

```
# Start  magnetometer logging at reboot.
@reboot /home/[USER ID]/run-mag.sh >> /home/[USER
ID]/projects/cronlogs/run.txt 2>&1
```

Use your favorite text editor (nano, vi, whatever…) to create a file in your home directory named run-mag.sh. Put the following line in this file:

```
~/rm3100-runMag/runMag -b 1 -j -k -A 10 -c 400 -U 140 -Z -S [USER ID] -O ~/rm3100-
runMag/logs &
```

Or (if you are using the Grape boards)**:**

```
~/rm3100-runMag/runMag -b 1 -M 23 -j -k -A 10 -c 400 -U 140 -Z -S [USER ID] -O
~/rm3100-runMag/logs &
```

Save this file, and then use the command to make the file executable:

```
$ chmod a+x run-mag.sh
```

It is important that this file’s name matches the one used in the cronlog file above.

Test this file by typing ./run-mag.sh at the command prompt. This should return the name of the log file being created.

If you just hit enter you should be returned to the command prompt.

You can view the output (if things go well) by entering:

```
$ tail -f  ~/rm3100-runMag/logs/[CURRENT Log File Name].log
```

This will show the output until 00:00 UTC when the runMag program switches to a new output file.

Testing and Operation

Some useful commands for debugging problems:

To see what your Pi or the cloud server thinks the UTC time currently is, enter:

```
$ date -u
```

Using this command from the rm3100-runMag directory allows output to go directly to the terminal window for use while testing and aligning:

```
$ ./runMag -b 1 -j -A 10 -c 400 -U 140 -Z
```

This version of the runMag command runs the program as it is normally used, sending output to a log file:

```
$ ./runMag -b 1 -j -k -A 10 -c 400 -U 140 -Z -S <user id>
```

Starting runMag runs the program in the background :

```
$./runMag -b 1 -j -k -A 10 -c 400 -U 140 -Z -S <user id> &
```

If you want to pull the task into the foreground (say to kill it with (control-C) just type:

```
$ fg
```

Use this command to see if the runMag program is running in the background:

```
$ ps -aux | grep runmag
```

Note: only one copy of the program can run at a time because the I2C devices can only be used by one process at a time.

Use this command to view the output of the program when it is running in the background:

```
$ tail -f logs/[CURRENT Log File Name].log
```

Note: If testing at around 00:00 UTC this command may appear to suddenly stop. This is because the program has rolled over to the next day's log file, and no new data is being written to the one that you are looking at. Just type <^C> (control C) to end the current command, and restart using the new log file's name.

Note: If while testing your setup, the program repeatedly reports an error and exits, be sure to completely power down you Raspberry Pi and then restart it. Repeatedly restarting the program without a proper reboot can leave the I2C drivers in an indeterminate state.

Note: If you see the following error message:

```
i2c_writebuf(): write(): Remote I/O error

i2c_read read value.: Remote I/O error

RM3100 REVID NOT CORRECT: RM3100 REVID: 0x0 <> EXPECTED: 0x22.
```

This means that you are not communicating with the magnetometer module. This could be because:

1. The PNI module is not (correctly) plugged in to Remote board.

   > Check the orientation.

2. The Remote board is not connected through using the CAT 6a cable.

   > Check the cables.

3. You are using a remote ‘Grape’ module and you have not included the ‘-M 23’ command in your runMag command line.

   > Check your command line.

4. (least likely, but possible) Your PNI rm3100 module is defective and needs to be returned.

   > return the module for replacement.