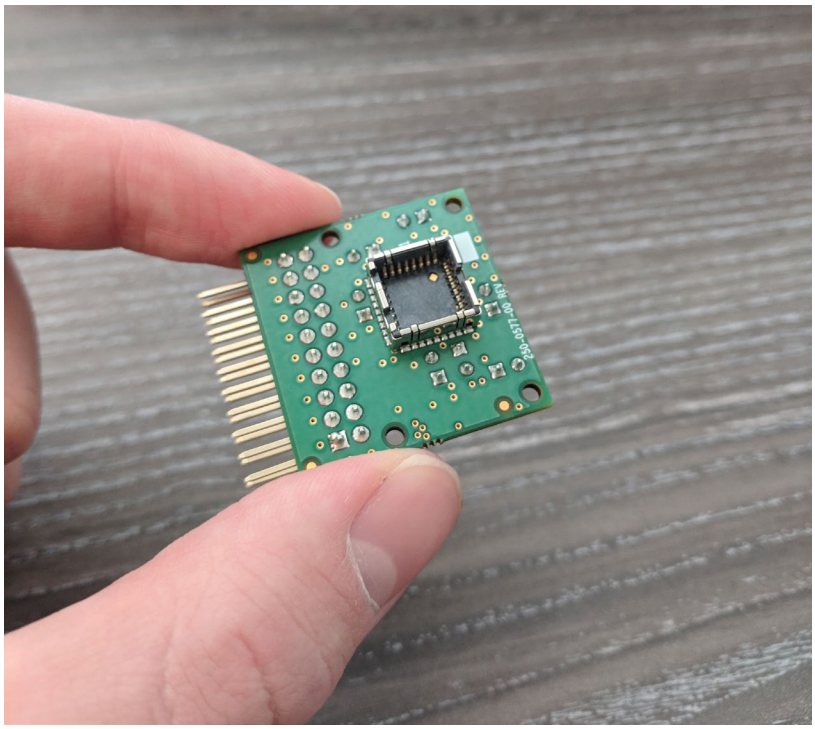
GETTING STARTED WITH THE RASPBERRY PI AND BREAKOUT BOARD V2.0

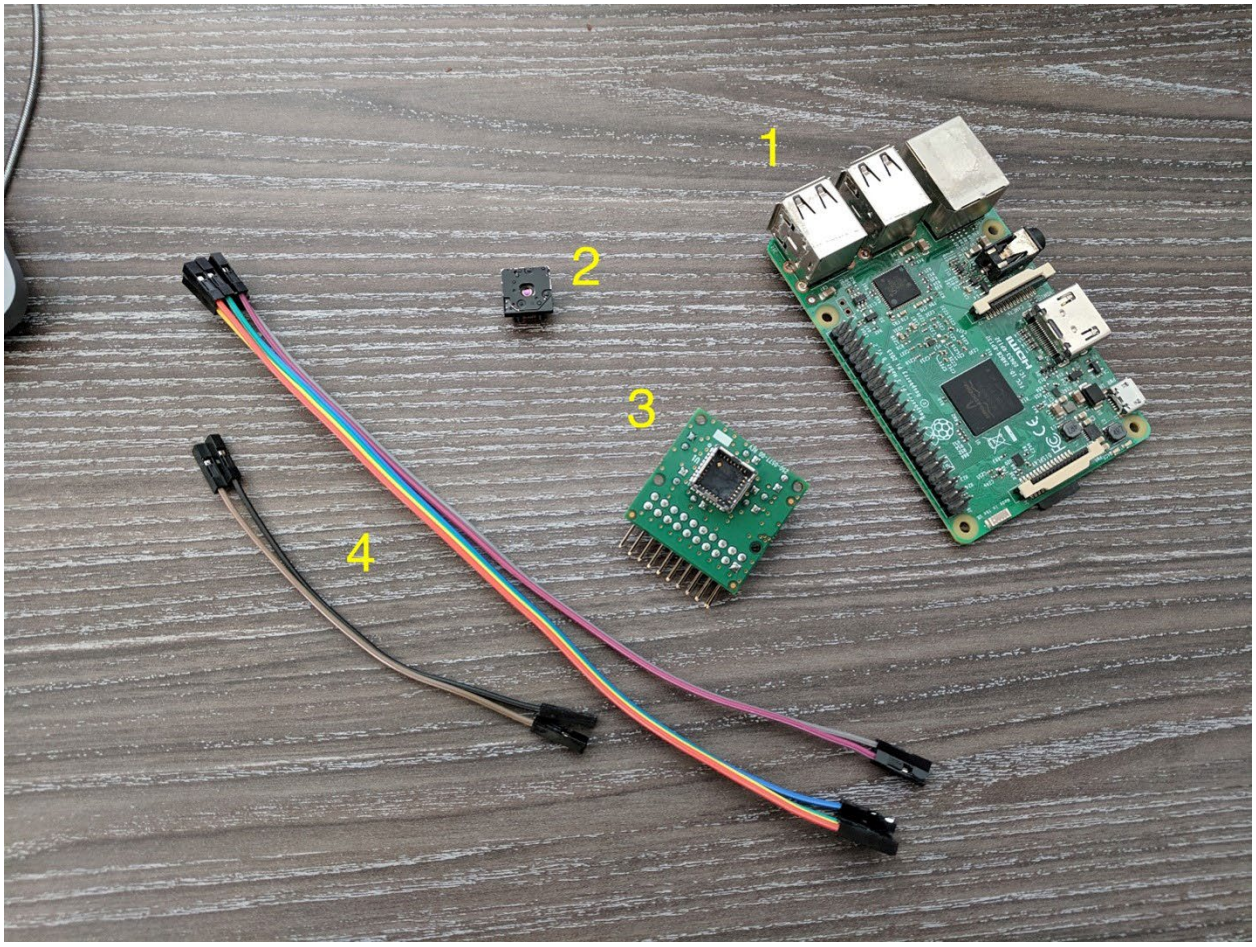
Disclaimer
This guide is for the newer breakout board (seen here). Earlier versions of the breakout board do not include VSYNC



Requirements

- A computer with the ability to read/write to MicroSD cards
    - A laptop using Linux was used during this guide.
- A RaspberryPi 2/3
    - You can find them here.
    - A MicroUSB to USB-A cable.
    - A MicroSD card with at least 8GB of capacity. MicroSD cards preinstalled with NOOBs can be found here.
    - Labelled as 1.

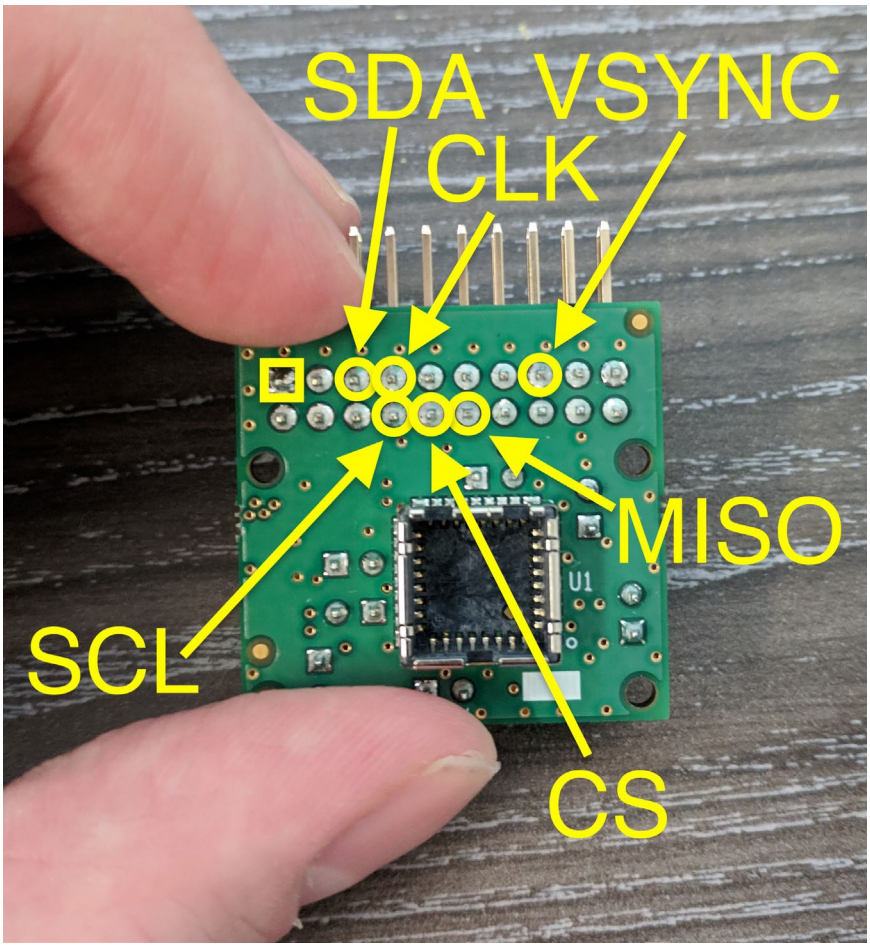
- A Lepton camera (2.X/3.X)
    - They can be found here. For a higher resolution and telemetry data, it is recommended that you use a Lepton 3.5.
    - Labelled as 2.
- A breakout board V2.0
- Details on the board can be found here.
    - Labelled as 3.
- Female-to-Female Jumper cables
    - Labelled as 4.
- Some additional software that can be found here.



Hardware

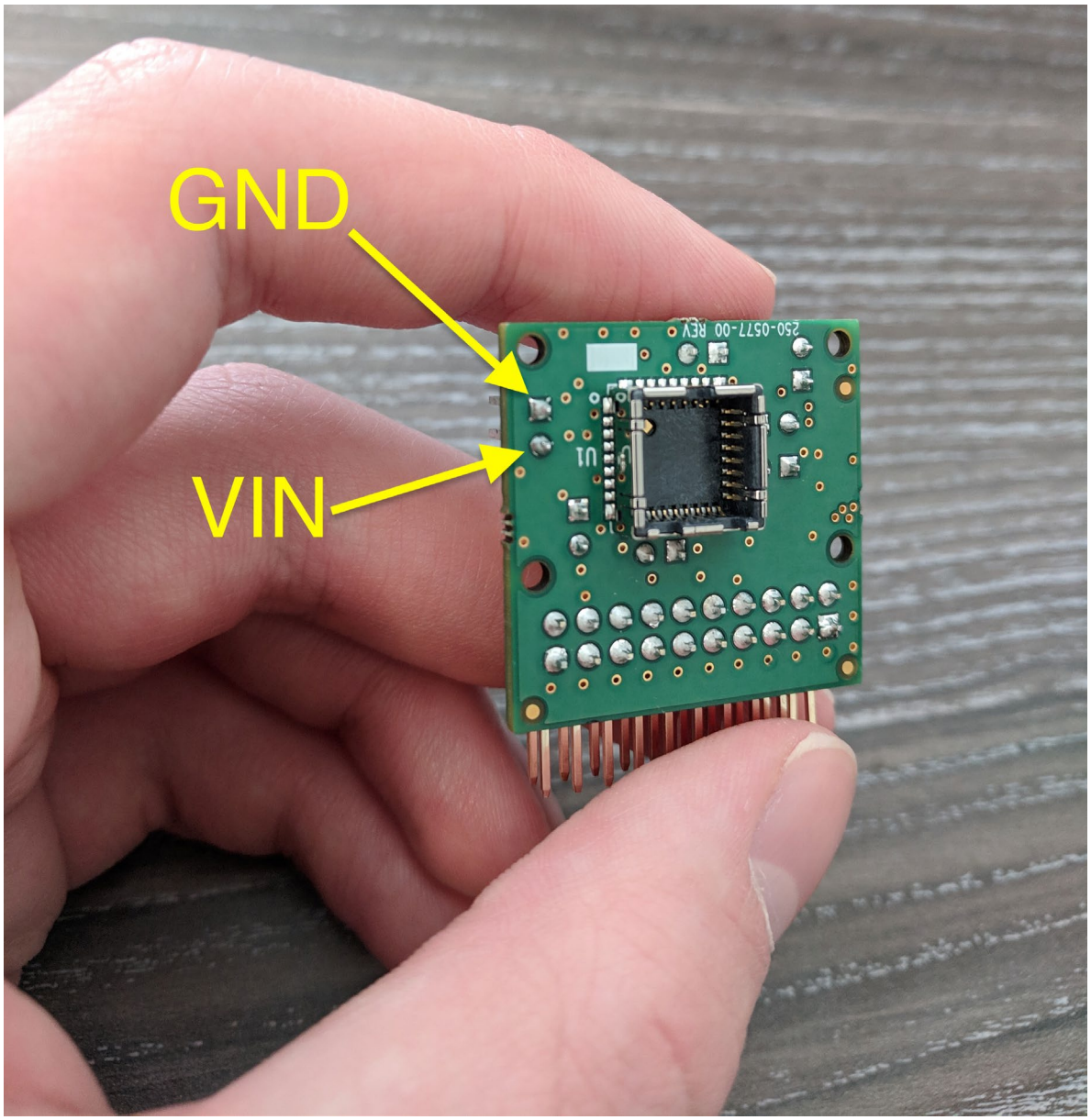
Attaching the breakout board to the RPi

You can find more information on the GPIO and the pinout of the RaspberryPi here. Attach (female-to-male) jumper wires between the following breakout board pins and the RaspberryPi board:



- (J2 Pin) -> (Proper name) -> (RPi connector pin)
- P8 -> SCL -> GPIO 3
- P5 -> SDA -> GPIO 2
- P12 -> MISO -> GPIO 9
- P7 -> CLK -> GPIO 11
- P10 -> CS -> GPIO 8
- P15 -> VSYNC -> GPIO 17

We will also need to connect power and ground to the board:



- (J3 Pin) -> (Proper name)
- P1 (Square pin) -> GROUND
- P2 -> VIN

Setting up the MicroSD

You have two options for setting up the Micro SD:

- Using NOOBs to get an image
- Building the image manually

NOOBS

You will need to install the latest Raspbian image to the MicroSD, which can be found here. The simplest way of installing the new image would be through the Raspberry Pi Foundation's NOOBs New Out Of the Box software.

A full guide to installing with NOOBs can be found here.

When installing through NOOBs, you will be walked through the installation and updating the raspberrypi to the latest version. For the purpose of this guide, the graphical interface wasn't used. To disable it you will need to adjust the settings:

```
Applications Menu > Preferences > Raspberry Pi Configuration:

Boot: "To CLI"
```

After adjusting this setting, you will need to reboot.

MANUALLY

If you want to install it manually, some instructions can be found here.

Software

Installing the Toolchain and building the Linux kernel from source

For the sake of convenience, I'll be covering compiling locally on the Raspberry Pi 2/3. For more detailed instructions, a guide to cross compiling or instructions for a different version of the Pi you can find the official documentation here.

Boot up the Pi. We will need access to a terminal and an internet connection. There are some dependencies, install them using the following command:

```
sudo apt-get install git bc
```

We will now acquire the sources. Warning: This will take some time. We will be including --depth=1, which will stop the repositories entire history being retreived and reduce the required storage space and download time significantly. For reference, the kernel version used for this guide was 4.14.71-v7+.

```
cd ~

git clone --depth=1 https://github.com/raspberrypi/linux
```

The new kernel will need to have some particular settings enabled for supporting DMA. The default config for bcm2709 should enable all the necessary options.

```
cd ~/linux

KERNEL=kernel7

make bcm2709_defconfig
```

Build and install the kernel modules:

```
make -j4 zImage modules dtbs

sudo make modules_install

sudo cp arch/arm/boot/dts/*.dtb /boot/

sudo cp arch/arm/boot/dts/overlays/*.dtb* /boot/overlays/

sudo cp arch/arm/boot/dts/overlays/README /boot/overlays/

sudo cp arch/arm/boot/zImage /boot/$KERNEL.img
```

We will need to ensure that the config.txt file correctly points to our new image. This can be done by opening the file located at /boot/config.txt:

```
cat /boot/config.txt
```

Ensure that the following line is present:

```
kernel=kernel7.img
```

After these steps are completed, reboot your Pi and ensure that the correct linux kernel is running:

```
sudo reboot

uname -r
```

Move example software onto Pi

Copy the RaspberryPi onto the Pi. If everything during installation was left to be default, it is recommended that you move it to the home folder. This guide will assume that the folder will be at this location:

```
/home/pi/.
```

Building the driver and applications

Now, from the RaspberryPi lets build the driver and test applications.

```
cd ~/RaspberryPi/lepton_module

make -C /lib/modules/`uname -r`/build M=$PWD modules

sudo make -C /lib/modules/`uname -r`/build M=$PWD modules_install
```

If the module fails to compile, it may be necessary to recompile the kernel, see above.

Setting up device tree and kernel module

We will next need to build the device tree overlay. Run the following command to compile flir-lepton-00A0.dts:

```
dtc flir-lepton-00A0.dts -o flir-lepton-00A0.dtbo
```

Ignore any warnings that come from compiling. You will need to move the .dtbo file into the overlays folder.

```
sudo cp flir-lepton-00A0.dtbo /boot/overlays/
```

You will then need to uncomment following values for flags in /boot/config.txt:

```
dtparam=i2c_arm=on
dtparam=i2s=on
dtparam=spi=on
```

This can be done by opening the file and ensuring there is no # in front of them.

```
sudo nano /boot/config.txt
```

This will enable i2c and spi after the Raspberry Pi is rebooted. Alongside uncommenting these, you will also need to add the following line:

```
dtoverlay=flir-lepton-00A0
```

This will enable the overlay you just built. You will need to reboot the RaspberryPi for the changes to config.txt to take effect.

```
sudo reboot
```

Once the Pi has rebooted, you will need to install the module. This will build the dependencies for all of the modules depmod -a and then enable the module we compiled modprobe lepton. You will only need to do this once:

```
sudo depmod -a

sudo modprobe lepton
```

We should be set! Lets test if this worked.

Testing the lepton camera

Once everything is connected, compile and run vsync_app under lepton_control

```
cd ~/RaspberryPi/lepton_control/

make

./vsync_app
```

The program should terminate quickly after performing various I2C commands. If it runs indefinitely, quit the program, disconnect the VIN and GND, and try again.

You can use the lepton_data_collector application to capture grayscale images from the video device. You will also need to make that too.

```
cd ~/RaspberryPi/lepton_data_collector/

make
```

- It consumes raw subframes via the V4L2 /dev/videoN device file (/dev/video0 by default) and extracts the pixel data.
- For Lepton 3.X (with the command-line argument -3) it assembles four subframes into a single larger video frame.
    - If you are using a Lepton 2.X, use the command line argument -2 instead.
- The image files are named after a prefix specified with the -o command-line argument, followed by a 6-digit (prefixed with 0's for smaller numbers) frame counter and a .gray extension.
- To reduce latency, it is a good idea to have it store frames into memory instead of to a flash device, so mount a tmpfs directory somewhere and prepend the path to the prefix.

Here is an example:

```
cd ~/RaspberryPi/lepton_data_collector/

sudo su

mkdir /tmp/capture

mount -t tmpfs tmpfs /tmp/capture

./lepton_data_collector -3 -c 50 -o /tmp/capture/frame_
```

When complete, there should be 50 images captured from Lepton 3.X (about 5 seconds worth at ~9 fps) named /tmp/capture/frame_000000.gray through /tmp/capture/frame_000049.gray.

These images can be viewed on a Linux system using the ImageJ Java application, available from here. From the File menu, select Import->Raw..., and set image type to 16-bit unsigned along with the width and height (80x60 for Lepton 2.X, 160x120 for Lepton 3.X) in the dialog box that appears after selecting the file name. The entire sequence can be placed into an image stack if the Open all files in folder checkbox is checked. Loading the files into an image stack makes it possible to produce a .avi movie from the frames using File->Save As->AVI... and setting the frame rate to 9 fps.

Troubleshooting

Before troubleshooting

Additional debug information can be retrieved by running:

```
dmesg | less
```

"I just can't seem to get the module to load!"

There might have been either an issue when compiling or the dependencies weren't correctly found. Try installing the lepton module manually again:

```
sudo modprobe lepton
```

If the lepton module can't be found, try running the following command to build the dependencies:

```
sudo depmod -a
```

Now, run modprobe again. If it outputs another error, try recompiling the kernel.