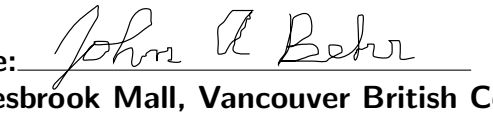
Upgrading TRINAT Control System and Optimizing Temperature

G. E. Claire Preston

Work Term Period: May 2015-August 2015
Program: Honours Physics Co-op, McMaster University
Project: TRINAT (TRIUMF's Neutral Atom Trap)
Supervisor: Dr. John Behr

Supervisor Signature: 

Location: TRIUMF, 4004 Wesbrook Mall, Vancouver British Columbia
Date Report: August 21, 2015

Contents

1 TRINAT Introduction

TRINAT, TRIUMF's Neutral Atom Trap, is one of the research projects at TRIUMF, Canada's national nuclear and particle physics laboratory. TRINAT operates a magneto-optical trap, which uses high-power lasers and a magnetic field to trap clouds of alkali metal atoms in a region of approximately a cubic millimeter in volume, at temperatures of down to a few hundred μK. Many different types of experiments can then be performed on these atoms.

1.1 Optical Molasses

The optical molasses method is used to cool the atoms down to sub-mK temperatures. As described in [1], atoms will absorb photons from laser beams shining upon them, and each absorbed photon will impart its momentum to the atom in the form of a scattering force, kicking it in the direction of the photon's motion. If one laser beam shines at the atom from each side, forces will be imparted on the atom from two opposite directions along a single axis. Atoms preferably absorb photons whose frequencies are closer to an atomic resonance frequency, corresponding to the energy needed for a specific electron energy level transition in the atom. If the original light is detuned to below the atomic resonance frequency for a particular transition, motion of an atom towards the light source will Doppler shift the frequency higher towards the blue, closer to the atomic resonance, such that the more photons from this beam will be absorbed. Meanwhile, the beam shining on the atom from the other direction will become red-shifted to a lower frequency since the atom is moving away from that source. This creates a larger damping force on the atom from the 'bluer' side since more scattering force is provided from that direction, slowing down the atom until the scattering forces are balanced when it absorbs equal numbers of photons from both directions because the frequency is no longer Doppler-shifted when it has stopped moving.

If three pairs of inward-shining laser beams are positioned such that they aim at a central group of atoms along three Cartesian axes, slowing in three dimensions can be achieved. As temperature is directly related to the speed of the atoms, this can achieve very low temperatures. It is almost like the atoms are moving in a molasses-like liquid, rapidly slowing them down due to a damping force. In certain atoms with more complicated electron energy level structures, other cooling mechanisms can also have an effect [1].

1.2 Magneto-optical Trap

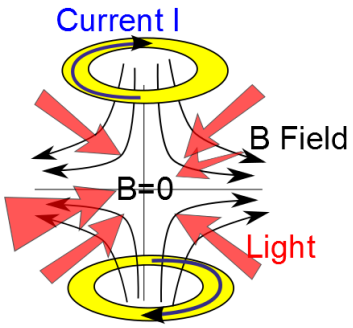


Figure 1.1: Magneto-optical trap

When the optical molasses technique causes accumulation of atoms within the laser beam intersection region due to cooling, this can be made into to a trap by choosing suitable light polarizations and adding a magnetic field gradient. Two metal coils with currents running

through them in opposite directions produce a quadrupole magnetic field as in Figure 1.1. We can consider the simple case of a total angular momentum J=0 to J=1 transition in an atom, which is similar to the J=1/2 to J=3/2 transition in the potassium atoms we use. At the center of the quadrupole field, B=0. The J=1 level has three sublevels, whose positions vary linearly with the atom's displacement close to this origin due to the Zeeman effect in presence of a magnetic field. The Zeeman shift of the resonance frequency corresponding to the transition causes preferential absorption of photons from the direction in which the atom has been displaced from the center origin. These photons are absorbed and the atom gets pushed back by the forces to the center, zero magnetic field region [1].

The TRINAT MOT is formed by first trapping atoms from either a dispenser or radioactive beam in the first trap, and then transferring them over to a second trap for the experiments.

1.3 Beta Decay Experiments

One of the major focuses of the TRINAT project has been on studying decays of unstable atoms to search for physics beyond the Standard Model of particle physics. The most recent experiments were run observing the decay of Potassium-37. A Potassium-37 atom decays into an Argon-37 recoiling nucleus, a positron and a neutrino.

$$^{37}K \rightarrow {}^{37}Ar + \beta^{+} + \nu$$

According to the Standard Model, only left-handed spin (a component of the angular momentum) neutrinos can be emitted from beta decay, so the neutrino would have to be left-handed, in that it has a 'left' direction of spin with respect to its momentum. TRINAT has been searching in recent years for evidence of right-handed neutrinos, which would spin in the opposite direction, to provide evidence for new physics.

2 New Trap Control System

2.1 System Overview

In short, a Raspberry Pi computer controls electronics to operate the atom trap, with new connections to a LabJack board and AB Electronics DAC-ADC Pi, as well as all the other old system components that are still used. It is connected to a new Point Grey Flea3 camera as well, such that it can digitally trigger it and detect the camera's signal when it has taken a picture. The camera images are read off and stored by a laptop, which also displays the images and their projection profiles as the images are captured so the trap status can be monitored, using Physica. The camera and trap control programs are written in C++, and shell scripts are used to coordinate the running of all the programs simultaneously.

2.2 Summary of Previous Work Done

My January-April term was spent on testing the capabilities of the new Point Grey Flea3 13Y3M-C camera for the replacement trap control system, and configuring and testing the hardware control capabilities of the Raspberry Pi B+ and Pi 2 running Real-Time Linux. For detailed info, refer to my previous work term report. A summary of important points is provided below.

The camera running Point Grey's experimental firmware to try and reduce the exposure time was found to have a minimum true exposure time, or a 'Frame Overhead Time' of 80 microseconds when the exposure time was set to 0 microseconds. This camera was found to be able to trigger continuously at intervals as low as 1.25 ms per image for small image sizes (around 200x200 pixels), though it is not well known if it can transfer the images this fast. Saving an image in memory takes a couple of milliseconds depending on the image size, and

saving to hard disk also only takes a couple of milliseconds. Programs have been written to capture images from software triggering, continuous image capture, and external triggering, and all work.

The optimized final system settled on was the Raspberry Pi 2 running a 3.18 real-time kernel compiled by Emlid developers for their Navio systems, which was freely available on the web. Running programs at highest priority on the real-time SCHED_FIFO schedule on this system, hardware GPIO digital signals are found to have an average latency per instruction of between 10-15μs, a delay that is small enough to be acceptable for our trap operation. Programming and timing tests for the GPIO pins and the AB DAC-ADC Pi were performed as well. Some research was done into the benefit of dedicating an entire CPU to the program operation, but when this was tested with the system under heavy load, it was found to not impact the latency of the high-priority tasks, but if benefit was proven this might be useful in the future.

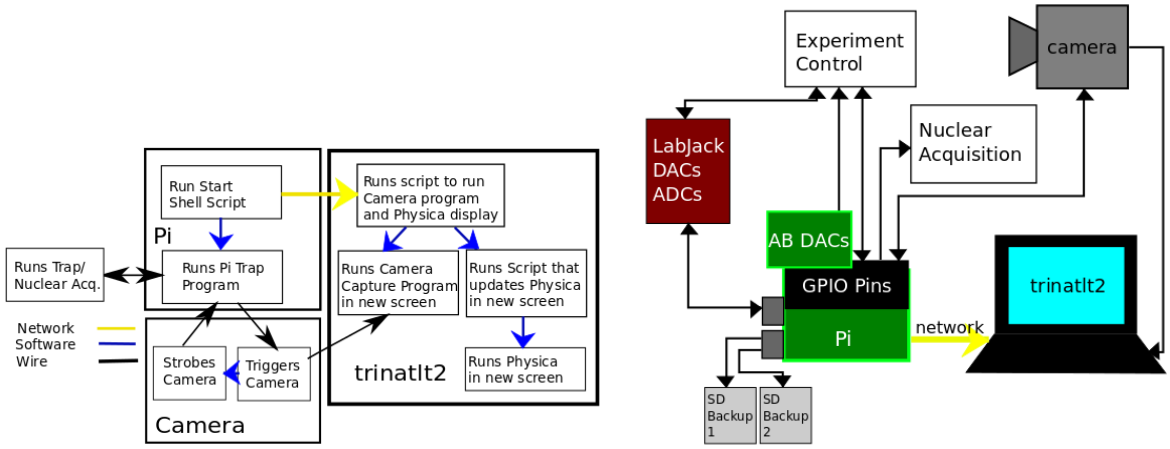
2.3 Experiment Control from Raspberry Pi

The main control system components that required programming include a Raspberry Pi 2, a laptop, the Flea3 13Y3M-C camera, a LabJack U3-HV board, and an AB Electronics DAC-ADC Pi board. Already in place is the Nuclear Acquisition system, and other electronic devices including lasers, acoustic-optical-modulators (AOM's), and other digital-to-analog converters (DAC's).

For operation, see Figure 2.1 for an overview of the new trap operation scheme, as I currently have made it to work, illustrating the coordination of the separate devices in the trap sequence and image capture. To start a trap sequence, one starts a shell script on the Pi (`StartSequenceDC.sh`). This shell script remotely connects to the laptop trinatlt2 and starts another script that will run the camera program and Physica real-time trap image and projection display. The script on the Pi then waits a few seconds and starts the trap sequence program, which operates external hardware connected to the Pi. Some of the signals go directly to Nuclear Acquisition to indicate what is occurring in the trap at different times. Others go into precision amplifiers made at TRIUMF, which are able to amplify voltage signals and provide current to operate various other devices for trap control, since the Pi's output current is limited to 16 mA per GPIO pin and 50 mA for all the pins together. Other signals go to DAC's and the camera.

On the laptop, the initialized script utilizes the Linux `screen` program to run two programs at the same time. The shell script creates one screen, or background terminal, on which it runs the camera program, cp5AsyncTriggerEx. It opens another that runs another script, which runs a Physica display window and displays images sequentially as they are saved in the folder after capture, giving an almost real-time display capability.

Reasons for setting the system up this way include that the Pi does not have enough power or memory to reliably run the camera in addition to everything else, and trying to do so may have incurred further latencies in the Pi system, throwing off timing in the operation. Using shell scripts and the `screen` program was seen at the time to be the best way to coordinate the operation of the 3 programs on the 2 different computers, but there may be better ways to do this.



(a) Software control between systems

(b) Layout of hardware system

Figure 2.1: Hardware and Software Control Overview

2.3.1 Hardware

The GPIO pins on the Raspberry Pi can be controlled with a variety of different libraries and native functions - we are using the bcm2835 library due to good real-time performace. Root user privileges are required to access the GPIO pins. Functions were written to simplify and clarify use - see Appendix A.

Currently free Raspberry Pi GPIO pins or those that are not really being used include 3 pins. Pins 3 and 5 are for the I2C bus, but this is not currently in use so these pins could be used. Pin 10 is not in use.

The LabJack board is a LabJack U3-HV and connects to the Pi via USB, controlled using the downloaded LabJack library. Functions were written to simplify the operation of the DAC's and analog-to-digital converters (ADC's) as outlined in Appendix A. The signals take a few milliseconds to respond to changes, so this is useful for slow control aspects of the trap. These are connected to the ports on the precision amplifiers with more filtering, since they are slower signals. The normal DAC's on the board can output voltages between 0-5V, while the extra connected Tick-DAC's have ranges of -10 to +10V.

The AB Electronics DAC-ADC Pi is controlled through the SPI (Serial Peripheral Interface) bus on the Raspberry Pi 2, using native Linux drivers. I have written functions to do this as well. Currently, on the two available SPI channels for the Pi, the two DAC's are in use, capable of changing voltage between 0V and 3.3V within a few microseconds, and connected to the ports on the precision amplifiers with less filtering to get a faster response.

Cables were ordered and made after planning GPIO pin layout as in Figure 2.2. Each labelled pin connects to a different control aspect, some through the precision amplifier, some directly to TTL boxes, some to the camera.

The Pi is set to back up to two bootable micro SD cards via USB every day using `crontab`. In the event of the card failure, either of these cards can be used as a replacement. Backups to other servers are also recommended. Details about the backing up is also located in Appendix A.

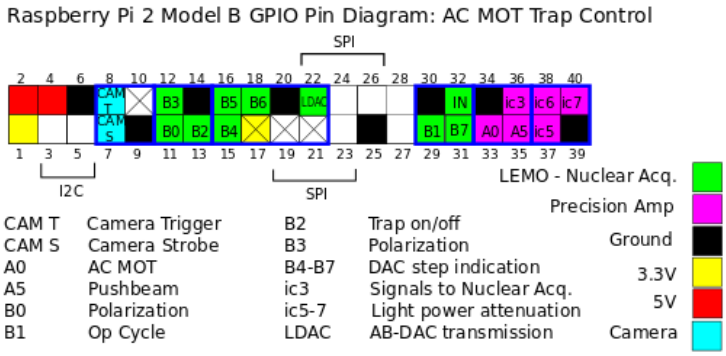


Figure 2.2: Layout of Pi GPIO pins setup

2.3.2 Software

The program for trap operation on the new system was rewritten from the original D90DC.C script on the old DOS computer system. For more detailed information see code documentation (Appendix A).

The main program is `TrapSeq_cp3_DC` for the DC trap. It reads parameters from a `params.txt` file on the Pi, providing values including pulse length, and space between pulses. This runs fine by itself when executed with `sudo` to allow access to real-time scheduling policies and usage of the GPIO pins, and can be run in conjunction with the camera by running `StartSequenceDC.sh`, which also runs the camera and Physica display programs on trinatlt2.

The pin and DAC names and functions, as well as various other definitions, are provided in the `testseq_definitions2.h` file. The functions for hardware operation are provided in `testseq_functions2.cpp`, and their declarations in the header file `testseq_functions.h`.

For the temperature measurements, three different versions of the code were made to vary the power, frequency, and length of cooling time. Another version was made for the AC magneto-optical trap.

2.4 Camera Control

Many weeks were spent during my January-April term implementating dynamic memory allocation and other requirements into adaptations of the included example C++ programs in the FlyCapture Software Development Kit provided by Point Grey, in order to control the camera. A lot of time in the second term was spent tailoring the programs to fit the functions as desired for trap operation.

2.4.1 Hardware

The camera is a Point Grey 1.3 Mega Pixel Flea3 13Y3M-C. It can be triggered via the GPIO 0 (I0) opto-isolated input connection with a Pi digital 'on' output pin signal and connecting to ground to opto-isolated ground (OPTO_GND). It can strobe after it has taken a picture, releasing a digital output signal that can be detected by a Pi input pin, via GPIO 1 (O1) opto-isolated output via a connected pull-up resistor.

The optimal value of this resistor was carefully tested to ensure that the 'high' voltage value falls above the input 'high' value for the Pi GPIO pins, 2.31 V, and the 'low' value falls below the input 'low' value for the Pi GPIO pins, 0.54 V [2], and settled to be $1k\Omega$. This gives a 'high' of 2.4 V and a 'low' of 0.5 V with a rise time of $70\mu s$ which is fine if the strobe length is set to above $100\mu s$. The maximum rate at which I have been able to get the camera to capture images is about 1 image per 1.25 ms so this is fine.

The fact that the circuits are both opto-isolated means they do not connect directly to any electronic circuits within the camera, but complete a circuit connected to an internal LED that then shines onto a photodiode that is connected to the camera electronics. This makes the camera safer against over-current or voltage connections and avoids ground loops.

In front of the camera there is a 1" 40mm lens 40mm from the sensor, and a 2" 630mm lens approximately 4" from that. In front of that there is a long-pass coloured glass filter that blocks room lights and only lets long wavelengths through, with a half-transmission point at 695nm. It is 92% efficient at the trap wavelength of 767nm. The camera was refocused in July and the scale measured, and is expressed later in the temperature section. The current total field of view of the camera at the position of the trap is approximately 9x7.5cm.

2.4.2 Software

For detailed programming information see Appendix A.

The camera connects to the laptop trinatlt2 via USB. When the trap sequence program is started via a `StartSequenceDC.sh` shell script on the Pi, this communicates with trinatlt2 over the TRIUMF network to initiate a shell script named `runCamandPhysica.sh` that runs the camera program as well as the Physica display of images as they are taken. The camera program is called `cp5AsyncTriggerEx`, based on Point Grey's `AsyncTriggerEx` example. This program is set up to read image settings from the `params.txt` file in the program folder, including gain, exposure time, and length of strobe, and then wait for a trigger signal from GPIO 0 before taking a picture.

One of the parameter options that can be set is `saveAfter`. If this is set to '1' the images will be stored dynamically in memory and then all written out at the end of program execution. This is useful if images must be taken in very rapid sequence. If it is set to '0' the images will each be written out after they are captured.

Actually storing the pictures in the hard drive vs memory only takes around 1-3 extra milliseconds depending on image size, with dynamic memory storage of the image also taking 1-3 ms depending on the image size. The images are stored in memory and written out as arrays. The program can be quit at any time by pressing the 'q' key if the camera has been properly set by the program into asynchronous trigger mode rather than continuous capture mode. The images are stored in the same folder as the program that is executing, and so they must be moved into a folder after the program is run or they will be overwritten in the next run.

The usage of the Linux `screen` program was utilized in order to be able to remotely start and run both the camera and Physica programs at the same time from the Pi computer. The `screen` program basically creates terminal windows for each new 'screen' that you assign a name to, and you can attach or detach from these terminals to monitor progress. If the `runCamandPhysica.sh` script is running, if one is in a trinatlt2 terminal window, they can enter the camera screen window by typing `screen -r camerawin`, reattaching to the 'camerawin' screen. To detach from a screen, either type `Ctrl-a d`, or into that respective terminal type `screen -d`.

2.5 (Almost) Real-Time Trap Display with Physica

Having a real-time monitoring system for the trap's status is important for the trap operation. My first attempt to produce this was with Physica, but this so far does not work very well and can definitely use improvement. Upon starting the `StartSequenceDC.sh` script from the Pi, a Physica window should appear on the screen of the computer from which you are ssh-ing, and begin displaying images as they are captured. For more detailed information about the script, see code documentation in Appendix A.

The Physica display script is initialized by the `runCamandPhysica.sh` script, and creates a new `screen` called `physicawin`, for the Physica window. In this window runs the script

`physicaImageDisplay.sh`, which detects if new images have appeared in the folder, and sends the commands for displaying them to Physica if they exist. Physica itself is run in another instance of `screen` called `physicascreen`. The screen `physicawin` delivers commands from four `.pcm` files to update the pictures.

The program displays a diffusion plot according to the image intensity values. From these values, the program attempts background subtraction using a background image taken `cp0.out` initially by the Pi program. On the top left is a projection of the vertical after background subtraction, and on the bottom right is a projection of the horizontal after background subtraction. These often seem to not be subtracted properly - the first suspicion may be if the trap light settings are not the same during the background picture as they are for later pictures.

The `physicaImageDisplay.sh` executes some `.pcm` files straight out but for others it reads in the commands one at a time. The reason for doing this was that I found when I tried to have it repetitively execute a script it would stop around the 70th cycle, claiming there were no more unopened units. This leads me to believe that the executed files are somehow remaining open after execution, which I could not manage to find a fix for. As a result, currently I am having the script read in lines individually from the `.pcm` files that update with each loop, with a delay between each command. This causes the `physicascreen` to often freeze, but it allows displays of images beyond the 70th.

If the Physica graphic window freezes, one must attach to the physicascreen by entering `screen -r physicascreen` and press `Ctrl-c` in order for the code to progress and keep showing pictures. One can then detach from the screen again by typing `Ctrl-a d`.

This display method works ok, but as stated above it often freezes, and when pictures are taken in rapid succession it cannot keep up with the rate of image capture and falls behind. Main suggestions for improvement of the Physica display operation include finding a way to close files after execution such that one can continuously execute a `.pcm` file instead of reading in individual lines from it, as well as not displaying every image, but only displaying the most recent image. Also displaying a total sum of images rather than each individual image might be more useful for experiments where parameters are not being varied.

My feeling is that it might be better to move away from Physica for this display purpose altogether as it seems very slow to handle the real-time display demand. Suggestions include writing a C/C++ other program using the OpenCV libraries to display images or if there is someone more advanced working on this, figuring out a way to piggy-back on Point Grey's FlyCap demo program in their real-time display capability using OpenGL.

3 Temperature Measurements and Optimization

Two key properties of the trap to be minimized are the temperature and the size. This is important for the momentum reconstruction to find that of the neutrino using the decay products. The location and initial momenta of the unstable atoms must be known as accurately as possible in order to obtain an accurate reconstruction.

3.1 Theory

3.1.1 Doppler Cooling Limit

Doppler cooling using lasers achieves an optical molasses state. For laser beams shining at a cloud of atoms from opposite directions, an atom at rest is equally likely to absorb photons from either direction, so the absorption is random, and each spontaneously emitted photon also goes in a random direction with respect to other emitted photons. The rate of increase of kinetic energy due to diffusion from absorption and spontaneous emission can be equated to the rate

of decrease in kinetic energy due to friction, such that the steady-state equation is

$$2E_r(2R_{scatt}) = \alpha < v^2 >$$

Where R_{scatt} is the scattering rate for each of the absorption and emission, E_r, the recoil energy, α is the frictional damping coefficient, and $< v^2 >$ is the time averaged square of the velocity. Since there is only one degree of freedom, $E_r = \frac{m<v^2>}{2} = \frac{k_B T}{2}$, where m is the mass, and T is the temperature. From Foot's derivations, we know expressions for R_{scatt} and α [1].

Putting these together, for Doppler limited cooling in one dimension, Phillips [3] gives us an expression for temperature

$$k_B T = -\frac{\hbar\Gamma}{4}\frac{1 + I/I_{sat} + \left(\frac{2\delta}{\Gamma}\right)^2}{\frac{2\delta}{\Gamma}} \tag{3.1}$$

Where Γ is the natural linewidth in Hz, I/I_{sat} is the intensity relative to the saturation intensity, and δ is the detuning from the resonance frequency of the particular transition.

Varying frequencies and powers can be entered into calculations accordingly to obtain predictions of temperature according to Doppler-limited cooling. It is important that this is only valid when I/I_{sat} is much less than 1. This is because at low intensity, successive absorptions are assumed to be unrelated, such that the Doppler shift due to the recoil velocity does not much affect the absorption probability. At high intensities, the absorption statistics are not Poissonian, so the variance in the amount of photons absorbed does not equal the mean, so certain assumptions made to obtain the above no longer apply [1] [3].

The natural linewidth was derived from the excitation lifetime $\tau = 26.34(5)ns$ [5], according to $\Gamma = \tau^{-1} = 37.986MHz$. The saturation intensity was obtained according to $I_{sat} = \frac{\pi}{3}\frac{hc}{\lambda^3\tau} = 1.75mW/cm^2$ [1].

3.1.2 Temperature Calculation

Dan Melconian, a previous PhD student in the group, did a few temperature measurements of the trap. From his doctoral thesis [4] we can derive the following calculation of the trap temperature. The trap ensemble approximately ballistically expands as a 3D Gaussian. The normal distribution width is therefore expected to grow according to

$$\vec{\sigma}^2(t) = \vec{\sigma_0}^2 + \frac{1}{2}\vec{v}^2 t^2$$

Where σ is the standard deviation of the fluorescence profile projection, v is the velocity, and t is the change in time. We can approximate the most probable speed as the average speed, where M is the atomic mass.

$$v_0 = |\vec{v}| = \sqrt{\frac{2k_B T}{M}}$$

In one dimension (along measurement axis)

$$T = \frac{M}{k_B}\frac{\sigma^2 - \sigma_0^2}{t^2}$$

$$\frac{\sigma^2(t) - \sigma_0^2(t)}{t^2} = \frac{\Delta(\sigma^2)}{\Delta(t^2)} = \frac{d(\sigma^2)}{d(t^2)}$$

$$T = \frac{M}{k_B}\frac{d(\sigma^2)}{d(t^2)} \tag{3.2}$$

This follows the same assumptions as the equation derivation summarized in the previous section, but here we make a measurement of the speed and can compare it to the Doppler-limited cooling prediction.

3.1.3 Number Of Atoms in Trap

An attempt to make a rough estimate of the number of atoms in the trap was made based on the camera info and the setup. This was done partially to investigate possible effects of different numbers of atoms on the temperature and trap size. This can be done with both the already-calibrated Electrim camera, and the Flea3, such that the two numbers can be compared.

The cameras have grey levels of 0-255. Each camera has a saturation level, the number of electrons per pixel that gives a 255 level. We assume that the number of electrons will be equal to the number of absorbed photons. We will call this value $n_{\gamma \cdot sat}$. If we divide this saturation number by 255, we get the number of absorbed photons per grey level per pixel, $n_{\gamma \cdot gl}$. Where we have a $Signal$ grey level and a corresponding background signal grey level $BGSignal$, the total number of photons that made the signal is

$$N_{\gamma \cdot det} = (Signal - BGSignal)(n_{\gamma \cdot gl})$$

The percent of total photons emitted by the trap that is expected to be detected is given approximately by

$$\left(\frac{\pi r^2}{4\pi R^2}\right)(\epsilon_F)(QE) = \epsilon d\Omega$$

r is the radius of the effective lens receiving area. R is the distance of the receiving lens from the trap. ϵ_F is the filter efficiency, and QE is the quantum efficiency at the wavelength of interest, in this case being 767 nm. ϵ gives the overall efficiency and $d\Omega$ gives the fraction of the full solid angle the photons go into. For r, each camera has two lenses in front of it. The lens closer to the trap has a focal length equal to the distance to the trap that makes the received light parallel, and the farther small lens of the camera focuses the rays onto the sensor. Because the light rays are made parallel by the first lens, one can approximate the effective lens receiving area as that of the farther small lens at the distance of the first lens from the trap, for large distances from the trap.

To find the number of atoms, we must take into account the number of photons each atom emits per second n_γ, and the amount of time we have been collecting photons t, in addition to the efficiency. At any given time only half the atoms are in the excited state at saturation levels, meaning the number of photons emitted per second per atom is half of the decay frequency, which is the inverse of the lifetime, $n_\gamma = \frac{\Gamma}{2} = \frac{1}{2\tau} = \frac{1}{2(26.34(5)ns)} = 1.898 \cdot 10^7/s$. Therefore, the number of detected photons is related to the number of atoms N_a by

$$N_{\gamma \cdot det} = (\epsilon d\Omega)(N_a)(n_\gamma)(t)$$

$$N_a = \frac{(Signal - BGSignal)(n_{\gamma \cdot gl})}{(\epsilon d\Omega)(n_\gamma)(t)} \tag{3.3}$$

Electrim Camera

In front of the Electrim camera is a Semrock LL01-780 filter, which allows 780nm light through at 95% at normal incidence, with the FWHM being only 2.7nm. This has been angle-tuned to accept 767nm light through. The best estimate of the filter efficiency was provided as around 92%, the similar to the filter in front of the Flea3, which is currently just a coloured glass filter and not as good.

It is known roughly that the saturation level $n_{\gamma \cdot sat} = 10^5$. For 255 grey levels, this corresponds to $n_{\gamma \cdot gl} = 392$ photons per grey level, ignoring the dark noise. $QE \approx 50\%$ at 770nm. The maximum signal after background subtraction seen in the trap for an unidentified run on August 12 was $5 \cdot 10^5$ for exposure time $t = 50ms$.

From the setup, we can determine the fraction of total photons collected to be

$$\epsilon d\Omega = \frac{\pi r^2}{4\pi R^2}(\epsilon_F)(QE) = \frac{\pi(1.25cm)^2}{4\pi(22cm)^2}(0.92)(0.5) = 3.71 \cdot 10^{-4}$$

Therefore

$$N_a = \frac{(5 \cdot 10^5)(392)}{(3.71 \cdot 10^{-4})(1.9 \cdot 10^7/s)(0.05s)} = 5.6 \cdot 10^5 atoms$$

Flea3 Camera

Given in the imaging performance manual, we obtain the absolute sensitivity threshold (γ)=44.13, which gives the minimum detectable irradiation in the mean number of photons that makes the signal-to-noise ratio SNR=1. But the saturation capacity (well depth, e^-) $n_{\gamma \cdot sat} = 10226$ [6]. From this, $n_{\gamma \cdot gl} = n_{\gamma \cdot sat}/255 = 40$, and this does not correspond to the 44.13 level. In the EMVA 1288 standards [7] that Point Grey uses to characterize their cameras, the absolute sensitivity threshold varies with responsivity, which depends on wavelength, so these numbers are probably different due to different sensitivty at different wavelengths. For our purposes, we chose to use the 40 number, but we disregarded the dark noise, as we were only looking for a order-of-magnitude estimate of the number of atoms. For reference, the dark current value was 26.26. At 770 nm, the quantum efficiency was about 36% [6]. Therefore, we determined that we should be able to find the number of atoms from the following calculation according to Equation 3.3.

$$\epsilon d\Omega = \frac{\pi r^2}{4\pi R^2}(\epsilon_F)(QE) = \frac{\pi(1.25cm)^2}{4\pi(63cm)^2}(0.92)(0.36) = 9.8 \cdot 10^{-5}$$

$$N_a = \frac{(Signal - BGSignal)(40)}{(9.8 \cdot 10^{-5})(1.9 \cdot 10^7/s)(t)}$$

3.1.4 Cross Section Calculation - Optical Density Effect on Temperature

It was thought possible that the high optical density could be causing heating through reabsorption and rescattering of photons randomly. This would cause noticeable effects on temperature and trap size with more atoms in the trap. The cross section is given as the intrinsic likelihood of one of these scattering events when photons hit particles as below [1]

$$\sigma_s = \frac{\lambda^2}{2\pi} = \frac{(767 \cdot 10^{-9}m)^2}{2\pi} = 9.36 \cdot 10^{-10} cm^2$$

The column density is found as $\rho_N = \frac{N_a}{A_{CS}}$, where A_{CS} is the cross-sectional trap area. The transmission is obtained from

$$T = e^{-\rho_N \sigma_s} \approx 1 - \rho_N \sigma_s$$

So the equation below gives the predicted number of photons that are reabsorbed. It is thought that at levels of around 10% this could start to produce a significant heating effect that would be seen in variation in the temperature and size for more dense vs less dense clouds.

$$\rho_N \sigma_s = \frac{\lambda^2}{2\pi} \frac{N_a}{A_{CS}} \tag{3.4}$$

3.2 Measurement Methods

3.2.1 Camera Scale

The scale of the camera was re-determined after refocusing on July 8, 2015, by imaging a 4 mm grid at the focal point with the previously mentioned coloured-glass filter installed to only let 767nm light through. The widths and heights of 8 boxes were averaged to find the scale to be $0.077 \pm 0.004 mm/pixel$. This is very close to the unfocused number given in the previous report as $0.078 mm/pixel$.

3.2.2 Laser Frequency

The frequency was determined from the frequencies of all other frequency-altering elements in the system.

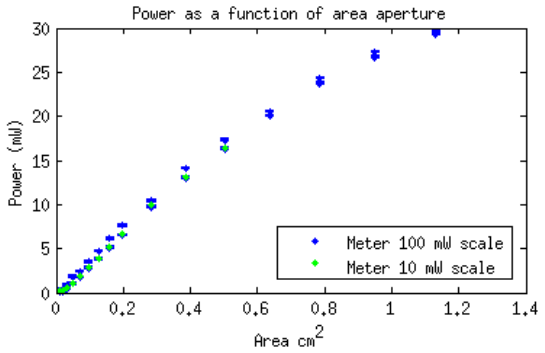
$$\nu_{trap} = \nu_{899} - |\nu_7| + 2|\nu_{CT}| - |\nu_{110}|$$

In the above expression, ν_{trap} is the frequency relative to the reference frequency of the final cooling light. ν_{899} is the frequency coming out of the 899, which is equal to the locking frequency ν_{lock} minus twice the frequency of AOM 9 ν_9, where ν_{lock} is the average of the frequencies $389.88MHz$ and $386.50MHz$, and $\nu_9 \approx 72MHz$, and is recorded each day as slightly different due to fluctuation. So $\nu_{899} = \nu_{lock} - 2|\nu_9|$. ν_7 is frequency change of AOM 7, ν_{CT} is the frequency change of the Crystal Tech, the value we are varying, and ν_{110} is the frequency of the 110. ν_{lock} was noted to have a possible fluctuation of $\pm 1.7MHz$, because the absorption line to which the laser was locked is a combination of several transitions from Potassium-41's s1/2 F=1 state to p3/2 states that are not well resolved. For the first runs on July 6, 7, 20, measurements of the frequency at certain voltages of ABDAC0 (DAC 7) were made in a range from 0.84-1.52V, corresponding to 60.05MHz-73.06MHz ν_{CT}, well encompassing the two ranges in the later experiments of 1.1-1.4V and 1.25-1.55V. This relation of frequency vs voltage was plotted and found to be solidly monotonic, such that later frequency values were interpolated from a linear model fit to this with a small vertical shift to obtain the frequencies.

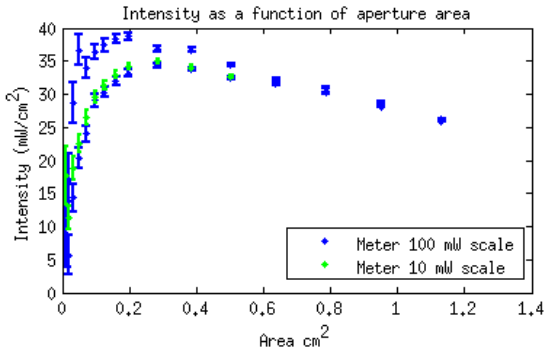
3.2.3 Laser Intensity

In order to find the theoretical Doppler-limited cooling curve (Equation 3.1), knowledge of the laser intensity in relation to the saturation intensity was required. The attenuator had 3 bits that could be activated for a total of 8 power levels. The bit attenuation levels were respectively 3dB, 6dB, 10dB, for the attenuator ZFAT-3610 from MiniCircuits.

We assumed the laser intensity had an approximately Gaussian profile. To measure the actual laser intensity affecting the trap, we decided to measure the power coming out of an aperture at varying diameters set in front of the laser beam. There was possibly hysteresis in the aperture, so measurement was made for both increasing and decreasing the aperture size, producing the plot in Figure 3.1b for intensity as a function of area. The intensity was chosen rather arbitrarily to be that at 5 mm diameter because this was assumed to be letting through all the light that would impact the trap and still be around the center of the Gaussian profile peak.



(a)



(b)

This means that the power levels are as follows with respect to percent of full power and saturation intensity $I_{sat} = 1.75mW/cm^2$.

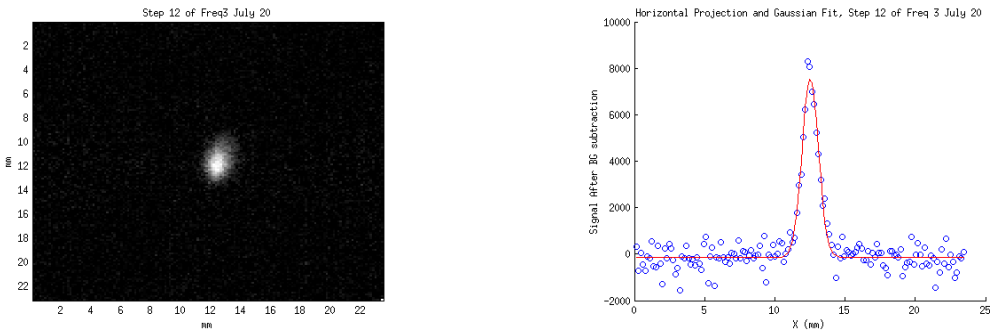
Atten. Level (bitwise)	0	1	2	3	4	5	6	7
Power (%)	100.0	50.1	25.1	12.6	10.0	5.0	2.5	1.3
Intensity (mW/cm^2)	35.3	17.7	8.9	4.4	3.5	1.8	0.9	0.4
I/Isat	20.2	10.1	5.1	2.5	2.0	1.0	0.5	0.2

3.2.4 Temperature

To determine the temperature, it is necessary to know the expansion rate with the trapping mechanism turned off. First the trap was established and cooled under different conditions of frequency, power, or cooling time, and then the trapping light was turned off and the trap allowed to expand. The light was turned on again to the same power and frequency setting as that for an initial background picture to take a picture after waiting a specified delay time.

For each step of the varying parameter, images were captured after 0, 1, 2, 3, and 4 ms of trap expansion. The duty cycle started with a background picture, then 8 or 16 steps varying frequency, power, or cooling time, imaging after 0 ms expansion with the trap light off, then 8 or 16 steps varying one of those parameters, imaging after 1 ms expansion with the trap light off, and so on. 21 or 22 images of each step were captured and summed, with the background image subtracted off of each one, to obtain a summed image of what the trap looked like at each step, as in Figure 3.2a. Each column was summed to create a horizontal projection of the fluorescence profile as in Figure 3.2b, to which a Gaussian could then be fitted. We chose to fit a 4-parameter Gaussian, adding a parameter for a vertical shift, such that the parameters are a, b, c, d.

$$y = ae^{-\frac{(x-b)^2}{2c^2}} + d$$



(a) Summed images at 0 ms expansion, 144 MHz, 10% Power for Frequency Run 2 on July 20

(b) Gaussian fit to horizontal sum projection

Figure 3.2

We can take the standard deviations of the Gaussian fits to the horizontal projections, and use the rates of expansion to determine the temperature as in Equation 3.2. Below is given an example fit of the $\frac{d(\sigma^2)}{d(t^2)}$ term for the expansion of the trap at 144 MHz and 10% power. We can obtain the standard error of the slope and use this as the relative uncertainty in our temperature calculation. This calculation can be repeated while varying different experimental conditions to find out how to minimize temperature, which is illustrated in the following section.

These fitting operations were carried out in MATLAB as this was my preferred platform. The Gaussian was an unweighted fit with a `NonLinearModel`, and the linear fit was done with a weighted fit (weighted according to the reciprocals of the variances of the standard deviation parameter c from the Gaussian fits) `LinearModel`.

This calculation, was checked in Physica to ensure that the slope values and standard error values were the same. Indeed the values were found to be consistent and the standard errors equal to the $E2$ parameter from Physica. The example code is located at

`trinat@trcomp:/home/trinat/cpreston/analysis/july20_tempPowerFreq2/power/temp_july20_2015_POWER/`

`rwprocp.pcm` and `physicafindtempcp.pcm` are the files used.

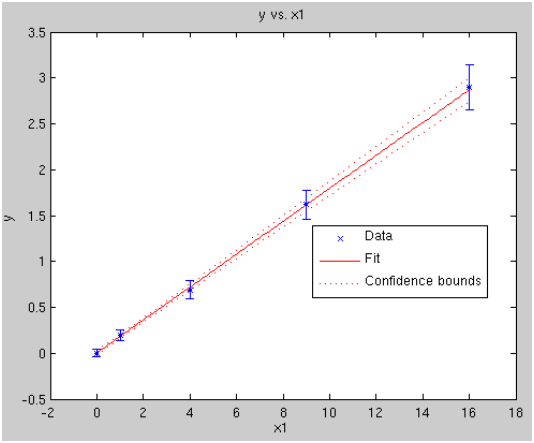


Figure 3.3: fitting $\Delta(\sigma^2)$ against $\Delta(t^2)$ for 0-4 ms expansion for 144 MHz at 10% power

3.2.5 Number of Atoms

To determine the number of atoms, we follow Equation 3.3. For comparison, we used the data from the runs that had occurred around a similar time to when the Electrim camera measurement was made, but we are unsure as to which run it corresponds with.

For the background-subtracted and 4 gaussian-parameter-fit projections, it was found that the baseline parameter d was not consistently 0, but varied between about -600 to 600 grey-level units between different runs, but more relatively consistent within each run. It is possible that some settings of the trap were not entirely the same between the different runs, changing the background level between the trapping and background image capture. It has been proposed that the laser power may have fluctuated over the hours in which the runs were done. Another possibility is that the trap light was not on for the entire duration of the image capture. A value of the average d vertical translation parameter was taken for each step at 0ms expansion for each run, multiplied by 152 pixels, and added to the overall sum of grey level units in attempt to adjust the tail of the Gaussian projections to actual zero, before calculating the number of atoms. If this was not done, for fewer numbers of atoms, often negative numbers of atoms were obtained from the calculation. The answers were only changed by approximately on average 20000 atoms, and as this was only an order of magnitude estimate, this seemed like a reasonable quick fix to better approximate the true background-subtracted signal. It would be better to scale the background image subtraction according to what is needed to make a zero background level, to accommodate for non-uniform intensity of the background, but there was unfortunately not enough time to perform this. The adjusted background-subtracted numbers for the signal strength for the trials in the same day as the Electrim camera observation were on average $1 \cdot 10^6$, $5 \cdot 10^5$, $1 \cdot 10^5$, $4 \cdot 10^5$, $2 \cdot 10^5$ (3 runs). These were all from summing together 22 images.

$$N_a = \frac{(Signal - BGSignal)(40)}{(9.8 \cdot 10^{-5})(1.9 \cdot 10^7/s)(0.001s)(22images)}$$

These return numbers that range from $9.8 \cdot 10^4$ to $9.8 \cdot 10^5$, in the range of the $5.6 \cdot 10^5$ atoms attained from the Electrim camera. These are used to express the temperature results in the charts below, where the number of atoms is taken only for the first data point, when the trap is still at its smallest size and has not yet expanded.

3.2.6 Optical Density

We are interested in an approximate idea of whether or not the densities of the trap are approaching levels where re-scattering of photons could be causing extra heating effects. We

will estimate this for a large and small experimental number of atoms in the trap. These values are also presented in the charts below.

For example, the July 31 Run 7 measure of trap temperature as a function of frequency, there are approximately 100000 atoms in the trap and a standard deviation of $0.46 \pm 0.01mm$, whereas Run 2 on the same day has closer to 30000 atoms, and a standard deviation of $0.42 \pm 0.01mm$. It was decided that the full width at half maximum $FWHM = 2\sqrt{2ln2}\sigma$ was a better approximation for the trap size rather than twice the standard deviation.

Going back to our optical density calculation, for the larger trap:

$$\rho_N = \frac{N_a}{(2FWHM)^2} = \frac{100000}{(2\sqrt{2ln2}(0.046cm))^2} = 8.5 \cdot 10^6 atoms/cm^2$$

$$\rho_N(\sigma_s) = 8.5 \cdot 10^6/cm^2(9.36 \cdot 10^{-10}cm^2) = 0.008$$

This is only an order of 1% of photons that are re-scattered. Repeating the calculation for 30000 atoms gets only 0.01% re-scattering. This does not seem like a level at which it might produce the large effects we are seeing between the large and small traps. For some of the larger traps, on the order of 500000 atoms, effects rise to approximately 4% re-scattering, which could possibly have more of an observable effect.

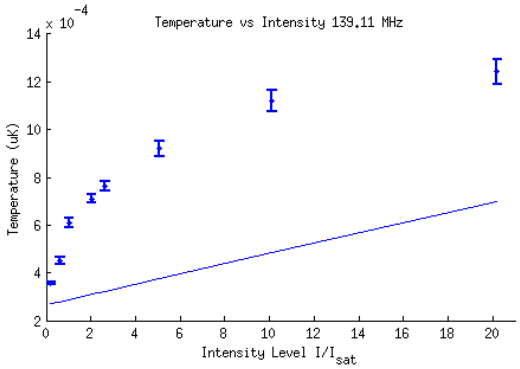
3.3 Temperature Optimization Results

3.3.1 Optimizing Power

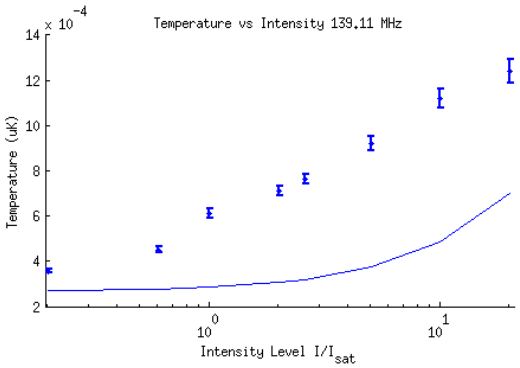
The temperature was determined as a function of varying the intensity from $0.35 - 35mW/cm^2$. This was one of the first tests done to observe the effect of power at the original frequency of 139.11 MHz. This was done in 8 power steps, imaging the trap after 0 ms expansion for the 8 power levels, then 8 at 1 ms, and so on, making a total of 5x8=40 steps, with 23 images per step. Below is a chart of the lowest and highest power results. The plot contains data points as well as the solid line Doppler-limited cooling prediction curve.

Temperatures when Varying Power at 139.11 MHz

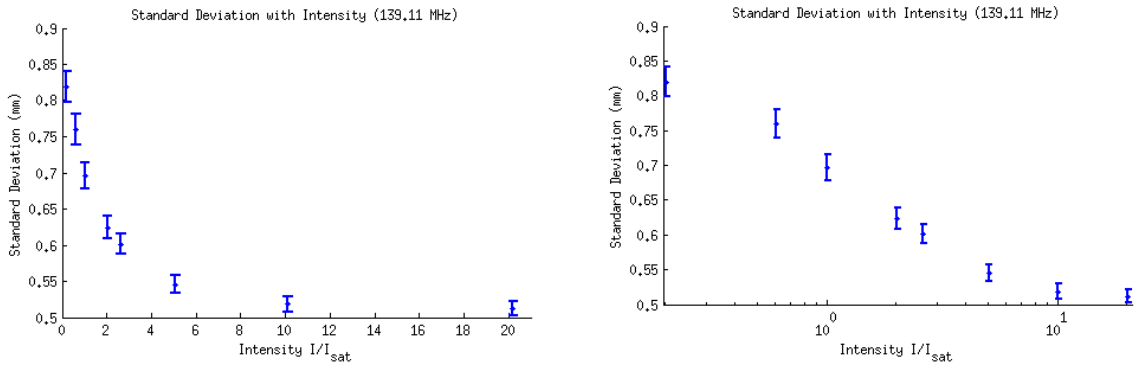
Date	Power	Temperature		Size	Number atoms (order mag.)	Rescattered Photons
		Experiment	Predicted			
	P/P0 (% full)	T (μK)	T (μK)	σ (mm)	Na (atoms)	(%)
July20_	0.01	531 +/- 11	271	0.82 +/- 0.02	94000	0.2
	1	2051 +/- 130	699	0.51 +/- 0.01	94000	0.6



(a)



(b)



(a) (b)

Figure 3.5: Measurements Optimizing Power

In the chart above and the plots it can easily be seen that for higher powers, the temperature is much higher, and much higher than the Doppler-limited cooling prediction for that particular power level and frequency. As power decreases, temperature decreases. But as power decreases, size also increases. This provides a problematic inverse relationship which is examined more in the next measurement.

3.3.2 Optimizing Frequency

The temperature was determined as a function of frequency by varying it within two ranges. The first two runs were done in a range of 132-142 MHz, and the others between 136-146 MHz. In a similar style to the previous, this was done by stepping through 16 frequencies after 0 ms expansion, then stepping through 16 frequencies at 1 ms expansion, then so on, for a total of 5x16=80 steps. The plots contain data points as well as the solid line Doppler-limited cooling prediction curve.

Temperatures when Varying Frequency and Power of Laser Light

Date_ Run #	Power	Experiment Minimum Temperature		Predicted T at this freq	Size	Theory Minimum Temperature		Number atoms (order mag.)	Rescattered Photons
	P/P0 (% full)	Freq (MHz)	Min. T (μK)	T (μK)	σ (mm)	Freq (MHz)	Min. T (μK)	Na (atoms)	(%)
July31_7	50	141.8	937 +/- 60	504	0.46 +/- 0.01	139.2	483.2	100000	0.8
July31_2	50 (low disp)	141.8	860 +/- 96	504	0.42 +/- 0.03	139.2	483.2	29000	0.3
July31_4	50 (low disp)	138.7	467 +/- 7	484	0.51 +/- 0.06	139.2	483.2	15000	0.1
July31_8	10	144.1	329 +/- 27	252	0.43 +/- 0.01	144.1	251.9	98000	0.9
July31_1	10 (low disp)	144.1	220 +/- 20	252	0.40 +/- 0.03	144.1	251.9	35000	0.4
July31_3	10 (low disp)	143.3	187 +/- 9	254	0.34 +/- 0.03	144.1	251.9	22000	0.3
July31_6	5	144.1	247 +/- 22	209	0.448 +/- 0.009	145.0	205.6	110000	0.9
Aug4_4	5	144.1	299 +/- 23	209	0.43 +/- 0.01	145.0	205.6	79000	0.7
July31_5	5 (low disp)	144.1	141 +/- 12	209	0.36 +/- 0.02	145.0	205.6	20000	0.3
Aug4_7	1	144.1	200 +/- 14	178	0.47 +/- 0.02	145.7	162.6	57000	0.4

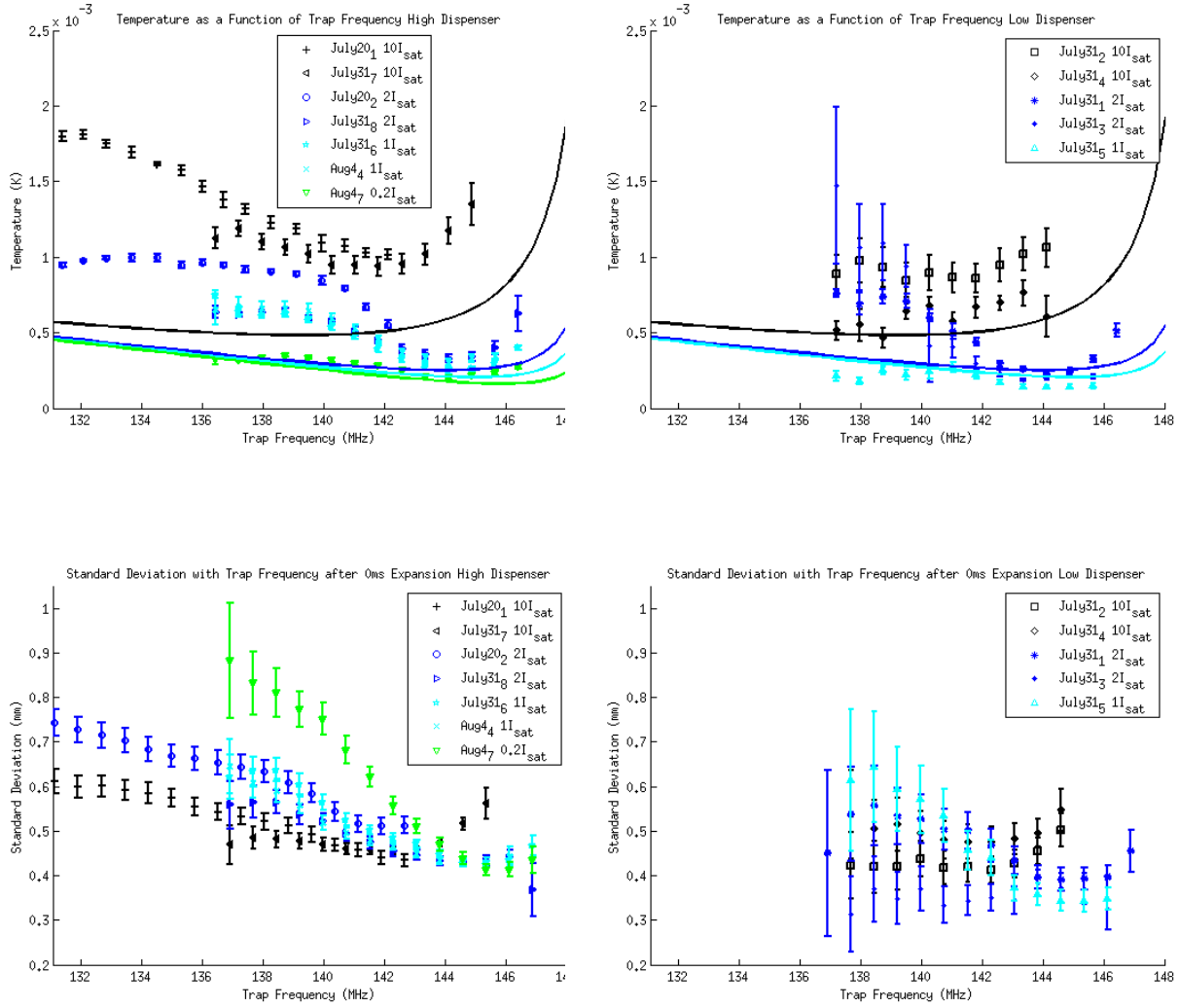


Figure 3.7: Measurements Optimizing Frequency and Power

See the chart above and plots for results when the dispenser was high and low (low disp. marks runs where the dispenser was low), at varying stepped frequencies and different power levels of light. In the chart, the experimental minimum temperature and frequency are just taken from picking the data point with the lowest temperature and stating its frequency and temperature. In the right column, the predicted minimum temperature and frequency give the predicted minimum temperature for that particular power level and at what frequency it is expected to happen for comparison.

It is evident that overall the higher the light power, the higher the temperature. From the plots of the standard deviations, one can gather that for lower powers, farther away from the frequency at which the minimum temperature is predicted, the size is larger, but close to the minimum predicted value, the sizes are all relatively small and similar.

Much difference is apparent between the runs with many ($\sim 10^5$) vs ($\sim 10^4$) fewer atoms. The temperatures achieved with fewer atoms are much smaller and very close to their Doppler-limited cooling predictions, and the sizes are smaller. With traps with more atoms, as the power increases, the distance of the experimental temperature from the Doppler-limited cooling prediction increases, as the trap evidently experiences some unknown heating effects, perhaps due to the large intensity of the light and re-scattering of photons. We can examine the optical density for this to estimate how much denser the trap is when there are more atoms and at higher powers, where the trap is of smaller size this still only gives us answers of around 1% (number of atoms

is estimated based on order of magnitude), which is not at a level expected to have a large effect.

For the two dispenser levels, the trap sizes seem to follow similar trends around the same value with respect to rising or falling at the same frequencies, though overall the smaller trap sizes are relatively smaller and a lot less well-defined as can be interpreted from the large error bars.

It is evident that the curves seem to be shifted to the left with respect to their prediction curves. This could potentially be due to the frequency being measured incorrectly. It was previously noted that the locking frequency could be inaccurate by up to 1.7MHz either way, so it very well could be that the frequencies should all be shifted higher by 1.7MHz such that they would then match better with the trends of their prediction curves.

3.3.3 Optimizing Cooling Time

There was interest in looking at trap's temperature and size as a function of how long it was cooled at different settings. First, to ensure the trap was small in size and that atoms were captured well, the original cooling scheme was performed, at half power, 139.11MHz, for 20ms. Then, the trap was cooled at a different frequency and power for varying periods of time before an image was taken. The duty cycle scheme is the same as with the frequency variation, but instead of varying the frequency in 16 steps, the second cooling time is varied in 16 steps, on a logarithmic scale from 10μs to 20ms. For most of the trials, a 500ms delay after the ACMOT section was removed, but for some trials this was added back in, and it was found that this made the trap have fewer atoms. Measurements were taken for both situations. The plot contains data points as well as the solid line Doppler-limited cooling prediction curve. In the legend, entries marked with a 'D' are where the 500 ms delay was left in.

Temperatures when Varying Cooling Times

Date_ Run#	Power	Experiment Temperatures			Predicted	Experiment Trap Size		Number atoms	Rescattered
			After 0ms cool	After 20ms cool	Temp	After 0ms cool	After 20ms cool	(order mag.)	Photons
	P/P0 (% full)	Freq (MHz)	T (μK)	T (μK)	T (μK)	σ (mm)	σ (mm)	Na (atoms)	(%)
Aug12/_1	0.05	144.7	853 +/- 15	259 +/- 12	210	0.434 +/- 0.003	0.497 +/- 0.002	661000	4.5
Aug12/_4	0.05	144.7	826 +/- 27	207 +/- 9	210	0.384 +/- 0.003	0.474 +/- 0.003	284000	2.1
Aug4/_5	0.05	144.7	785 +/- 35	226 +/- 7	209	0.374 +/- 0.007	0.448 +/- 0.009	121000	1.0
Aug12/_2	0.05 (delay)	144.7	935 +/- 27	167 +/- 10	210	0.359 +/- 0.009	0.49 +/- 0.02	87000	0.6
Aug12/_5	0.05 (delay)	144.7	631 +/- 43	206 +/- 15	210	0.36 +/- 0.01	0.45 +/- 0.02	52000	0.4
Aug12/_7	0.025	144.7	722 +/- 36	235 +/- 9	189	0.43 +/- 0.01	0.422 +/- 0.008	86000	0.8
Aug12/_6	0.01	144.7	808 +/- 31	240 +/- 7	178	0.472 +/- 0.005	0.458 +/- 0.004	263000	2.1
Aug12/_3	0.01 (delay)	144.7	758 +/- 34	211 +/- 9	178	0.45 +/- 0.01	0.47 +/- 0.02	53000	0.4
Aug4/_6	0.05	142.8	812 +/- 18	331 +/- 13	232	0.473 +/- 0.004	0.472 +/- 0.004	312000	2.4
Aug4/_8	0.01	142.8	813 +/- 19	281 +/- 8	209	0.544 +/- 0.005	0.473 +/- 0.004	330000	2.5
Aug4/_9	0.01	142.8	777 +/- 22	270 +/- 9	209	0.542 +/- 0.006	0.473 +/- 0.004	295000	2.2
Aug4/_1	0.5	139.1	986 +/- 31	993 +/- 34	483	0.497 +/- 0.004	0.494 +/- 0.004	313000	2.2
Aug4/_2	0.5	139.1	1015 +/- 43	1011 +/- 43	513	0.498 +/- 0.004	0.496 +/- 0.004	302000	2.1
Aug4/_3	0.05	139.1	940 +/- 37	511 +/- 11	288	0.600 +/- 0.007	0.487 +/- 0.004	262000	1.9

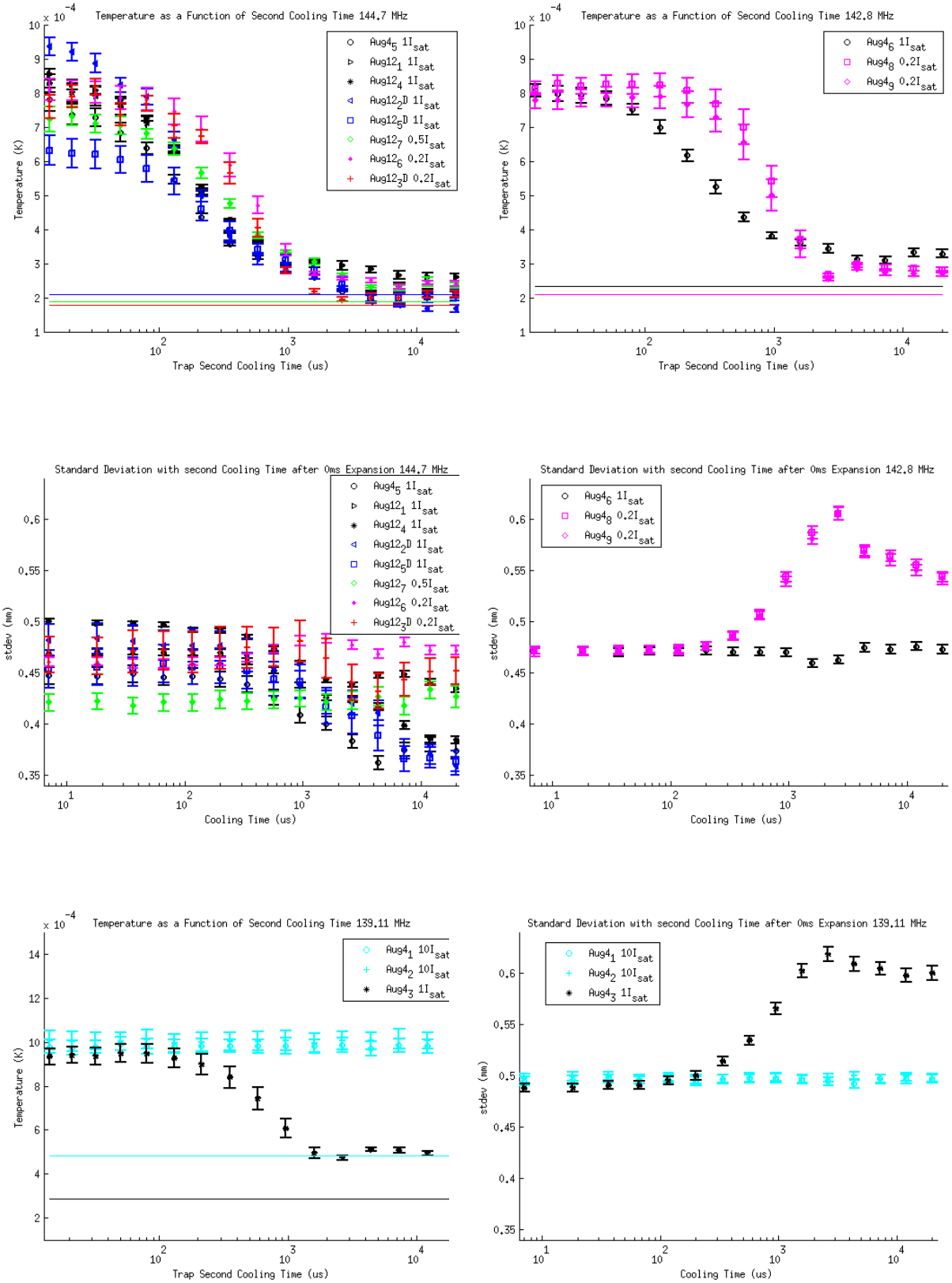


Figure 3.10: Measurements Varying Second Cooling Time

It is evident that all the trap settings take a similar amount of time to adjust to the new temperature setting. This is on the order of 2 ms, which at first seemed really long seeing as the decay time of the atoms is 26 ns. One of the possible explanations is that the atoms must

move in order for the Doppler cooling to take effect, and they are moving very slowly, at around 0.5m/s at these temperatures, making a delay before they are actually affected.

The 139 MHz run was intended more as a 'control' at the beginning to observe how the trap appeared when the new duty cycle was introduced but the settings not changed. As expected from the previous frequency measurements, with the lower power, as this is far away from the predicted minimum temperature frequency, the trap size grows a lot when it adjusts to the new power level.

The 142.8 MHz measurement was an intermediate measurement, and indeed much better temperatures were shown here. It is observed that changing the settings for the different power levels causes the atoms to adjust at different rates even though at a similar time, with the 1% power adjusting faster than the 5% power. I found this odd, as they are initially at the same temperature and so should be moving at the same speed and be equally affected by the Doppler cooling, but perhaps this has something to do with the fact that fewer photons are being absorbed. Again, it is found that for the lower power, where the frequency is far from the predicted minimum temperature frequency, the size grows, where the 5% power is close to the frequency at which its minimum is predicted and so it actually stays the same size

The most interesting runs are at 144.7 MHz as these produce the lowest temperatures. Looking at the 5% (1 Isat) runs, one can see no consistent correlation with number of atoms and size or temperature. This is predicted to be at a frequency close to the predicted minimum for many of these power levels, such that the sizes do not get larger as predicted by the frequency measurements. For some of the lower power measurements it is even seen to get smaller, but this is not universal. These findings do not refute findings from the frequency measurements, as they also found that the sizes did not vary very much close to the frequency at which the minimum temperature was predicted, but farther away from these predicted minima the size can be seen to be following the same trend.

The small kinks where the temperature seems to go low then higher again after adjustment, or high then lower again, along with the sizes, remain unexplained. One potential reason for this is imbalance in the laser beams or slight misalignment.

3.3.4 AC MOT

All the above experiments were done with the direct-current (DC) MOT, but we also had to test the alternating-current (AC) MOT. The AC MOT was tested using the `TrapSeq_cp3_AC` code. First images were taken of both the DC MOT and AC MOT to test that both were working as expected at the original frequency and power. Then a few settings were tried at which better temperatures had been observed in the DC MOT were tried, only imaging the AC MOT. The exposure time of the camera was also varied. The background image was not taken with the correct power, so other runs were done after unplugging the laser beam that pushes the atoms into the trap to get a background from that. Displayed below are background-subtracted images of the qualitative appearance of the AC MOT compared to the DC MOT.

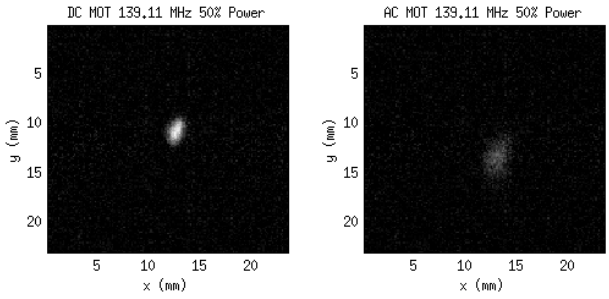
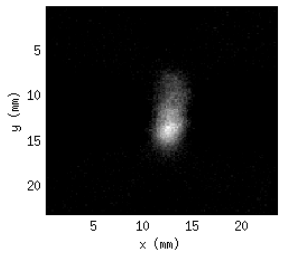
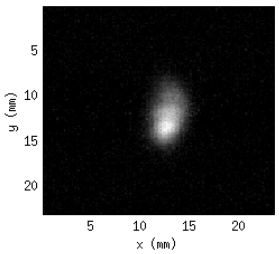


Figure 3.11: DC MOT and AC MOT under original 139.11 MHz 50% power at 1 ms exposure, sum of 626 images



(a) AC MOT at 144.7 MHz 5% power at 100 ms exposure, sum of 630 images



(b) AC MOT at 144.7 MHz 10% power at 200 ms exposure, sum of 70 images

Figure 3.12

Visually, it is apparent that the different frequency and power settings did not help the AC MOT as much as had been hoped to achieve a smaller size. The intensities cannot be compared due to different exposure lengths and summing different numbers of images.

3.4 Discussion

3.4.1 Achieving Minimum Temperature

It is evident from the power and frequency variation measurements that smaller temperatures and sizes can be achieved when the laser power is lower and the frequency is close to the Doppler predicted minimum. If I were to readjust the trap I would pick a frequency of 144.7 MHz (AB DAC 0 set to 1.45 V) and a power level of 5% (AOM power set to 6). It must be taken into account that at lower powers the atoms fluoresce less and the signal is harder to detect, and there wasn't a large temperature difference between 1% and 5% power.

It was considered possible from the results that having a larger number of atoms for frequencies farther from the predicted minimum could be making the size of the trap larger, and the temperature larger, but closer to the predicted minimum, this was not very apparent.

The optical density is probably not high enough, based on the number of atoms predicted, to explain why for higher powers of light the temperature is so much larger than the Doppler-limited cooling prediction, while it is much closer to the prediction for fewer atoms.

3.4.2 Error Analysis

The background subtraction is an evident large source of error. The fact that the vertical shift of the tail of the Gaussian projection fit varied between -600 to 600 between different runs for sums of 22 images indicates that this is not being done properly. Due to time constraints and the fact that this was only noticed when calculating the number of atoms near the end, this could not be well accounted for. It is possible that the laser power could fluctuate over the 10-20 minutes a run lasts, or that the power for the background image was incorrectly set, or changed over the course of the exposure time, but the second situation is assumed unlikely, as all images were taken at half power and this was not changed throughout the different revisions of the program. This could have a large effect on changing the number of atoms, as well as possibly changing the Gaussian fits, which would affect the temperature measurements. Because the shapes of the experimental curves were observed to follow the expected trend, it was assumed that the temperatures could still be held to some degree of accuracy. The fact that they were measured within the same run, for which the d vertical shift parameter was fairly consistent indicates that within the run it might have less of an effect than comparing between runs.

Another source of error previously mentioned is that the locking frequency apparently can vary up to 1.7 MHz either way, which could shift our Temperature vs Frequency curves closer to their shapes as seen in the Doppler prediction curves.

It is possible that due to other systematic errors unaccounted for and the background subtraction issue that the error bars should be larger. However, this is still a good qualitative indication of the relative temperatures of the trap under different settings.

4 Future Plans

The lowest temperatures were obtained with the lowest power currently available, 1% of full power=35 mW/cm^2. The theoretical Doppler cooling limit is approached as power goes lower. Therefore, it would be potentially useful to try and obtain a lower temperature by further attenuating the laser power.

The measurements obtained for temperature while varying frequency were done over a large range and it would be useful to do higher resolution measurements over a smaller frequency range around the Doppler minimum predicted values respective to the power levels.

There was concern as to whether the 1 ms exposure and laser light shining on the atoms for capturing the image could be affecting the trap, in causing them to be retrapped during the 1 ms. It would be good to shorten the exposure time to gain a better approximation for the trap appearance directly after the cooling. This would potentially bring in problems of getting into the dark noise region, so must be fiddled with.

For running the AC MOT, the trap is allowed to expand first after the DC MOT for 2 ms. It would be very useful to know how large the trap gets during this time. Instead of looking at only the standard deviations of the projections after 0ms, it would be a good idea to average the size over the first 2 ms and use this number in the next step calculations for the AC MOT.

References

[1] Foot C.J. *Atomic Physics*. Department of Physics, Oxford University. Oxford University Press, 2005.

[2] GPIO Electrical Specifications: Raspberry Pi input and output pin voltage and current capability. Mosaic Documentation Web. August 20, 2015. <http://www.mosaic-industries.com/embedded-systems/microcontroller-projects/raspberry-pi/gpio-pin-electrical-specifications>

[3] Phillips W.D. Laser Cooling and Trapping of Neutral Atoms. Laser Manipulation of Atoms and Ions. `Course CXVIII, Proceedings of the International School of Physics <Enrico Fermi>`. Italian Physical Society. Varenna on Lake Como, 1991. pp. 289-296.

[4] Melconian D.G. Measurement of the Neutrino Asymmetry in the Beta Decay of Laser-Cooled, Polarized 37K. Simon Fraser University, 2005. pp. 103-107.

[5] Wang H. et al. Precise determination of the dipole matrix element and radiative lifetime of the 39K 4p state by photoassociative spectroscopy. Phys. Rev. A 55(3), R1569(R). American Physical Society, 1997. 10.1103/PhysRevA.55.R1569.

[6] Flea3 USB 3.0 Digital Camera: Imaging Performance Specification Version 2.0. Point Grey, 2014. <http://www.ptgrey.com/support/downloads/10110>

[7] EMVA Standard 1288: Standard for Characterization of Image Sensors and Cameras 3.0. European Machine Vision Association, 2010. <http://www.emva.org/cms/upload/Standards/Stadard_1288/EMVA1288-3.0.pdf>

Trap Code Documentation: Appendix A to Work Term Report

G. E. Claire Preston

Work Term Period: May 2015-August 2015
Program: Honours Physics Co-op, McMaster University
Project: TRINAT (TRIUMF's Neutral Atom Trap)
Supervisor: Dr. John Behr
Location: TRIUMF
Date Report: August 21, 2015

Contents

1 How to read this document

This document is an informal, more detailed reference of how my codes for trap, camera, and Physica display operation work. There are many other codes I wrote, which can be referred to according to last term's report, but these ones are the ones I have worked on this term specific to the trap operation. Each section lists the directory in which the files can be found and the files referred to in that section. If there are many files in the same folder, the directory will be listed under the section title and the files under the subsection titles. There will be general notes about the specified sections of codes, as well as function names and descriptors of how they work. If you are looking for info on the operation of a particular function, or part of a file, have the file in front of you while consulting the description.

2 Shell Scripts

2.1 Brief intro to `screen` program

The four different programs (Pi Trap Sequence, Camera, Physica) are run simultaneously on different computers by using the command-line program `screen`. If this is not installed it can be obtained through `sudo apt-get install screen`. It basically opens an instance of a terminal window with a name you provide, and using this name identification you can connect to or disconnect from this terminal instance window while a program is running in it. You can send commands to different screens. Important commands to know include:

`screen -ls` lists all active screens running on the system
`screen -r 'insertscreennamehere'` 'attaches' to a screen with the identified name, or connects to it so it is active in your window
`screen -d` or `Ctrl-a d` 'detaches' from a screen so it is still running in the background but you are not monitoring it from your window
`screen -S 'insertscreennamehere'` creates a new screen with indicated name and you are attached or connected to it immediately
`exit` or `Ctrl-a k` terminates the screen you are in
For more info, type `man screen`.

2.2 Pi

`pi@trinatrpi2:/home/pi/RunTrapSequence_cp/`

2.2.1 Start Trap Sequence

`StartSequence.sh`
This starts the whole trap system operation including the scripts to run the camera and Physica on trinatlt2. It utilizes the `screen` program on both the pi and trinatlt2. A `screen` is created to connect with trinatlt2, to which it makes an ssh connection, and then it starts the script `runCamAndPhysica.sh` on the other computer. It allows 6 seconds for this to start and then starts the local trap sequence program.

2.3 trinatlt2

`trinat@trinatlt2:/home/trinat/flycapture/flycapcode/src/cpAsyncTriggerEx/`

2.3.1 Runs Camera and Physica Screens

`runCamandPhysica.sh`
This closes previously running `screen` and program instances, then starts a `camerawin` screen, on which it runs the `cp5AsyncTriggerEx` program. It creates a `physicawin` screen on which it runs `physicaImageDisplay.sh`.

2.3.2 Runs Physica Display

`physicaImageDisplay.sh`
This manages the running of Physica and its updating as new images are written to the folder for the display. It creates a new `screen` instance on which Physica itself is run, `physicascreen`. It retrieves the x and y dimensions and binning number of the image from `params.txt`. It tells Physica on `physicascreen` to run the `physica_defaults.pcm` script, and then sends it the width and height values, then runs the `physica_initvalues.pcm` script. It checks in a loop to see if a background image exists, and if so, executes the commands on the other screen in `physica_bgpic.pcm` to read in and display the background image. It then checks in a loop for the image of the next index to see if it exists, and if it does it reads the commands in `physica_displayspics.pcm` into the `physicascreen` separated by a delay of 10 ms each. This is to try and keep Physica from freezing, though it still often does, in which case one must attach to the `physicascreen` and press `Ctrl-C`. After, it increments its index and looks for the next image to appear before updating again.

3 Trap Sequence on Raspberry Pi

`pi@trinatrpi2:/home/pi/RunTrapSequence_cp/`

3.1 Makefile

`Makefile`
The name of the program desired to be compiled can be put in the `EXEC` variable, and then `make` should compile it.

3.2 Main Trap Sequence Files

`TrapSeq_cp3_DC(.cpp)` `TrapSeq_cp3_AC(.cpp)`
These are the main trapping sequences. They run the hardware with real-time scheduling priorities. They utilize the bcm2835 library to operate the GPIO pins and SPI bus for the AB Electronics DAC-ADC Pi. They read certain labelled parameters from the `params.txt` file located in the same folder to set certain system parameters.

3.3 Trap Hardware and Software Definitions

`testseq_definitions2.h`
This is an important file outlining the given names for referral of hardware including names and reference of GPIO pins, LabJack DAC's and ADC's, and AB Electronics DAC's. Most of these values are just names assigned to numbers, such that in the code, the name can be passed to functions, but really it just passes a corresponding number. The reason the numbers are in `CAPITALS` is that they are absolute constants across the program and this is important to keep track of.

3.3.1 GPIO Addresses

The first items define the GPIO pin address numbers. The numbers the pins are assigned to such as `PIN26` is defined as `RPI_V2_GPIO_P1_26`, come from definitions in the bcm2835 library, which is used to operate them. These pin numbers correspond to their layout on the pin diagram, numbered in order from 1-40. Further down in the definitions file, the new pin definitions are given names according to their function, such as `AOM_BIT0` is defined as `PIN37`, and these can be changed fairly easily and it is clear to see which pin corresponds to what function.

3.3.2 LabJack Definitions

The LabJack DAC's are given numbers for referral, assigned to their names, such as `LJDAC0` is defined as 0, such that LabJack functions can be passed this name and end up with actually just a number reference. This is intended for better clarity in reading and changing the code. The file also defines LabJack values such as the number of AIN (Analog Input) channels, and whether quick sampling and long settling time should be set for the AIN's.

3.3.3 AB Electronics DAC's

In the same way, numbers for the AB DAC's are given for referral. Constants relevant to the AB DAC's are also set.

3.3.4 Level definitions

The file defines the numbers (1 or 0) for the 'high' and 'low' states of the GPIO pins, as well as 'on' and 'off', and the polarization sigma states of 'plus' and 'minus'. These are expressed as binary values, but can be passed to functions and used in comparison with their definition names to clarify the code when reading and changing. Different power levels to the AOM for power of light attenuation for the cooling are also given as `AOMPOWERLEVEL0` to `AOMPOWERLEVEL7`, which

can be passed to the function `setAOMPower(int)` to set the power level according to the three bits. See function for more info.

3.3.5 Other

There are a few values relevant to setting the priority of the program that should remain constant. There are a few initial state definitions of what the polarization state should start as. There are numbers used to define the line numbers of the elements in `params.txt`. To change this, you can add more definitions on the bottom of the `enum` list and they will take on the next sequential number indices. These values are used in conjunction with `readParam(...)`, so see that function for more info.

3.4 Trap Includes

`testseq_includes.h`
This file just includes all the necessary headers for compiling the programs.

3.5 Trap Functions

`testseq_functions.h` `testseq_functions2.cpp`
These files contain functions that are used in the trap sequence. These include functions for implementing real-time scheduling policies, timing pauses, running GPIO's, running LabJack DAC's and ADC's and running AB Electronics DAC's.

3.5.1 Timing

Uses `clock_nanosleep` and the `CLOCK_MONOTONIC` to make a delay of a passed integer number of microseconds. This is known as a relatively high accuracy timing method to use in software.

`void msPause(int msToPause)`
Uses `clock_nanosleep` and the `CLOCK_MONOTONIC` to make a delay of a passed integer number of milliseconds. This is known as a relatively high accuracy timing method to use in software.

`double getTimems(struct timespec* pclockVal)`
Accepts a pointer to a `timespec` struct that has been used with a `clock_nanosleep` delay function and returns the time in milliseconds.

3.5.2 GPIO Pins

`bool setGPIOOutputs()`
This uses the bcm2835 library functions to set up required pins as outputs.

`bool setGPIOInputs()`
Uses the bcm2835 library function to set up required pins as inputs.

`bool out(short pinNum, short outType)`
Sets the state of the specified pin to either `HIGH` or `LOW`, which represent values 1 and 0 as specified in the definitions file, `testseq_definitions2.h`

`void setDAC_BITS(unsigned int dacVoltage)`
Sets GPIO output pins `DAC_BIT0` to `DAC_BIT2` to bitwise number to be sent to nuclear acquisition. The integer number entered is translated bitwise into the 3 bits being on or off (ie. sending

number 4 turns on DAC_BIT1 and turns off the other two bits)

`void setSIGMA(short sigmaPM)`
Sets polarization state to PLUS or MINUS as numerically defined as 'high' or 'low' in the definitions file testseq_definitions2.h. This controls two pins, the SIGMAPM pin and the SIGMAPM_TO_NUCLEAR pin. This sends a signal to nuclear acquisition as well as flipping the polarization.

`void setAOMPower(short AOMPowerLevel)`
Sets attenuation level of AOM controlling laser cooling light by controlling 3 GPIO pins corresponding to the three bits of attenuation. This can be passed previously defined levels in the testseq_definitions2.h file, from numerically defined AOMPOWERLEVEL0 to AOMPOWERLEVEL7, corresponding to activating bits 0-2 for the attenuation, corresponding respectively to 3dB, 6dB, and 10dB to change the power level.

`bool in(short pinNum, bool inType)`
Can be used to test state of an input pin, between 'high' or 'low'. Pass it the pin name, corresponding to a number, and the HIGH or LOW value you would like to test for. It returns 'true' if the state and the testing state match.

3.5.3 SPI

`void setupSPI()`
Sets up the SPI bus with the bcm_2835 library for use of the AB Electronics DAC-ADC Pi. For accessing different SPI channels, chip, or different SPI clock speeds (it is currently set to the max 20 MHz that the AB Electronics chip can read), this function should be edited. To change any of this refer to the bcm2835 library files and documentation. Here, it selects the first SPI chip, CS1, corresponding to the DAC's, but one could also select CS0 instead, which would activate the ADC's. There are only 2 SPI channels at a time available on the Pi.

`bool outABDAC(short dacNum,float dacVoltage)`
Sets either ABDAC0 or ABDAC1 to the float voltage as prescribed, between 0 to 3.3 V. It is a 12 bit DAC, and requires a 16 bit signal to set the voltage. In the testseq_definitions2.h file the gain on the AB DAC chip is set to 0, but it can also be set to 1, but one has to be careful as the chip should not be made to output more than 3.3 V. This function attempts to safe-check that it is not set higher than this. Some bitwise operations are performed on the voltage passed to this function before it is sent to the chip via the SPI bus.

`void closeSPI()`
Closes the SPI bus and frees up the pins.

3.5.4 LabJack Board

These functions all use the LabJack library, and code is largely copied and adapted from the LabJack U3-HV example programs. For further documentation, see the LabJack documentation and example code provided with the library.

`bool setupLJ(HANDLE* phDevice,u3CalibrationInfo* pcaliInfo,int* isDAC1Enabled)`
Sets up the LabJack board using the first found LabJack. If multiple LabJack boards are used, this will have to be changed. It then gets the calibration info for the board, then configures all inputs and outputs, and sets up a writemask for the various devices.

`int configAllIO(HANDLE hDevice, int *isDAC1Enabled)`
Configures all inputs and outputs on LabJack board, requiring sending and receiving 12 bytes. This configuration sets DAC 1 as enabled, and sets FIOAnalog and EIOAnalog so that they can work with the Tick DAC's. The command is sent and the response is read back and checked for errors, as is the case for all other LabJack commands as well.

`int setupWriteMaskLJ(HANDLE LJBoard, u3CalibrationInfo* caliInfo,int isDAC1Enabled)`
Sets up the writemask for writing to the different bits corresponding to ports of the LabJack board. Sets writemask (devices to write to) of FIO, EIO, CIO, receives and checks response.

`int outLJDAC(HANDLE LJBoard,u3CalibrationInfo *caliInfo,int isDAC1Enabled, int dacNum, float dacVoltage)`
Used to set voltage value to LabJack DAC (one of two on the board itself, not the Tick DAC's). The function must be passed the board ID, calibration info, whether DAC 1 is enabled, which DAC to select, and the float voltage value to set it to, between 0-5 V. Again, it sends a series of bytes and receives a series of bytes that it checks for errors.

`int checkI2CErrorcode(uint8 errorcode)`
Checks I2C bus error code for communication of LabJack board with attached Tick DAC's.

`int inLJADC(HANDLE LJBoard,u3CalibrationInfo *caliInfo,int isDAC1Enabled, int adcNum, double* pADCVal)`
Reads LabJack ADC value, depending on input of board ID, calibration info, whether or not DAC 1 is enabled, the number of the ADC to check, and a pointer to a double in which to store the read-off ADC value. In a similar way to the output, a string of bytes is sent and received and checked for errors. The double voltage reading is stored in the double pointer.

`int outLJTDAC(HANDLE LJBoard, u3TdacCalibrationInfo* pTdacCaliInfo,int dacNum,float dacVoltage)`
Used to set LJTDAC to a float value between -10 V and +10 V. It must be passed the board ID, a pointer to the calibration info for the particular Tick DAC, the number of the Tick DAC, and the desired float voltage value. This function has been written in a way that seemed convenient in which before each command is passed, the calibration info is read and set again, intended to be more flexible for changing the configuration of the Tick DAC's plugged into different sockets. So first the calibration info is obtained and then the actual command is sent to the correct Tick DAC.

`void closeLJ(HANDLE LJBoard)`
Closes the USB connection to the passed LabJack board.

3.5.5 Setting Priority and CPU Affinity

`void stack_prefault(void)`
Sets the size of the stack in memory to avoid the stack getting full and having to reset in the middle of the program operation, incurring latency.

`int setupPriority()`
Sets up the real-time `SCHED_FIFO` scheduling priority for the program so that it executes above all other user tasks at a set priority. Then it locks program into memory to avoid it being switched out to swap area on hard disk, incurring latency. Then it pre-faults the stack using the previous function.

3.5.6 Quit Key Exit Loop

`int isQuitPressed()`
As Pi seems to not wait for user prompt when looking at input keypresses by default, this was the only function needed to check if 'q' and 'Enter' had been pressed during program execution, indicating a quit. If it detects that yes 'q' and 'Enter' have been pressed, it will return 1, otherwise it will return 0 or -1 for error.

3.5.7 Reading and Writing Parameters File

`void readParam(std::ifstream*,unsigned int*, unsigned int)`
Reads an input parameter from the `params.txt` file, as an input file stream, according to its set line number, which is defined in `testseq_definitions2.h`. It stores the value using the passed `pValue` pointer.

`void readParam(std::ifstream*,double*, unsigned int)`
Same as previous function but for reading a `double` from the file (overloaded function).

`void writeOutParam(std::ifstream*,std::ofstream*)`
Writes out parameters with date and time of run to the log file to keep track of what parameters were on each run. Is passed a pointer to the input file stream `params.txt`, as well as an output file stream, the log file that it is written into along with the date and time.

`char* getDtTm(char*)`
Retrieves date and time to be written to log file.

4 Special Main Sequence Files: Temperature Measurements

`pitrinatrpi2:/home/pi/RunTrapSequence_cp/Temperature_JulyAug2015/`

4.1 Varying Frequency

`StartSequence_SweepFreq.sh`
Starts the frequency varying program and camera and Physica remotely on trinatlt2.

`TrapSeq_cp3_temperature_sweepFreq(.cpp)`
This is the same as the main sequence file except that it is set to vary the frequency with each step, as well as vary the expansion time with the trap light off with each step. Hardware wise, `DAC_BIT0=PIN15` has been switched to control the absolute on/off power of the trapping light in order to be able to fully turn it off to allow expansion. All the `setDAC_BITS()` instances have been commented out. There are a total of 80 steps, with 16 frequency steps, and 5 expansion time steps. The frequencies are set as 16 steps between `ABDAC0SweepMin, ABDAC0SweepMax`, incrementing with each step. The expansion time delays are read out from a file named `trapTempDelay.txt`, with 80 entries, determined by the set number `TempIntervals`, which are read from 1-80 and then back to the beginning again. The power is determined during the cooling and can be changed and recompiled. I set this up so that it varies the frequency with each image, but keeps the expansion delay the same, so it takes images at the 16 different frequency steps at 0 ms expansion, then 16 images at 1 ms and so on.

4.2 Varying Power

`StartSequence_SweepPower.sh`
Starts power varying program and camera and Physica programs remotely on trinatlt2.

`TrapSeq_cp3_temperature_sweepPower(.cpp)`
This is the same as the above file except there are only 8 power levels instead of 16 frequency steps, making a total of 40 steps, and they vary from 0-7.

4.3 Varying Cooling Time

`StartSequence_SweepCoolTime.sh`
Starts cooling time varying program and camera and Physica programs remotely on trinatlt2.

`TrapSeq_cp3_temperature_sweepCoolTime(.cpp)`
This is the same as the frequency varying program but instead of varying the frequency, it varies a second cooling time. The atoms are still cooled at the original half power and frequency for the 20 ms, and then cooled at another variant amount of time at a different frequency and power. Like the expansion delay, the cooling time is read from a file called `trapTempCoolDelay.txt` sequentially in a loop.

5 Camera External Trigger Program on trinatlt2

`trinat@trinatlt2:/home/trinat/flycapture/flycapcode/src/cpAsyncTriggerEx/`

This program uses an external trigger and strobe of the camera in conjunction with the Raspberry Pi to capture, dynamically store in memory, and save images. It utilizes a few parameters for the camera from `params.txt`. It uses the FlyCapture library included in their software development kit as well as functions I wrote into my own `cp1ImageMethods.h` and `.cpp` files.

5.1 Makefile

`Makefile`
Typing `make` should suffice to recompile the program.

5.2 Main camera program

`cp5AsyncTriggerEx(.cpp)`
This controls the entire camera operation and sequence. One can press 'q' to exit at any time during capture process. It reads camera parameters including strobe length, exposure time, and width and height, binning factor, from a `params.txt` file, with line numbers for each variable defined in the `enum paramLineNums` enumeration in the program file, with names such as `STROBELENGTH_LINENUM`, in a similar way to how the parameters file is read for the Pi trapping program, using a function called `readParam(...)`. *Note that you can read the `README.txt` file in this folder for more info on the restrictions on the parameters. that It writes out the parameters, the date and time, into a log file. It tests to make sure it can write files in the folder, and then sets up the camera, connects to it, and gets its information. Then it sets the camera settings desired, including mode, format, image sie, exposure time, and gain. It puts the camera in trigger mode, and sets the time it will wait for an image to 500 ms. It creates a `Actual_data_listcp` item to store the image objects dynamically in memory in a linked list, as well as a `datavalues_itemcp` called `currentitem` to store each temporary image before putting it in the list. It creates a Point Grey `Image` object to store the captured image in. It then sets the terminal termios settings so that it will not wait for a user prompt (Enter), but will continue looping, when it checks if 'q' for quit has been entered. This means that it can continue to take pictures and check for a keypress without the user having to press Enter. A few preliminary images are captured and not saved to ensure that the new camera settings fully take effect (ie

new exposure time etc.). Then the real images are taken in a loop waiting for external trigger signals or 'q' to be pressed. The image is stored in the data values item, and then either saved or stored in the data list. At the end of the cycle, the camera is disconnected.

5.3 Camera Written Classes and Functions

`cp1ImageCaptureMethods.h cp1ImageCaptureMethods.cpp`

5.3.1 Data values item class

`class datavalues_itemcp`
This class participates in the dynamic memory allocation process for obtaining images from the camera `Image workingImage` object and stores it in an object with required info that I have designed to store temporarily in memory before either being added to a data list in memory as a `Actual_data_listcp` item, or written out to file. One important member of this class is the `datavalues_itemcp* nextcp`, which points to the next datavalue array item in the list when this object is added to the data list for dynamic memory allocation. This is an element of a linked list - allowing us to know what element is the next image in the list.

`datavalues_itemcp(const datavalues_itemcp&)`
Copy constructor to create a copy of an object, and is `private` to prevent this being done to protect the proper structure of the object and ensuring it is initialized and destroyed in memory properly.

`datavalues_itemcp(unsigned int datasize_cp,int addBin,int width, int height)`
Constructor for datavalues item. Creates new item according to number of entries `datasize_cp`, number of pixels each in x and y to bin into 1 pixel `add_Bin`, and width and height of image. Assigns new array of provided size to `actualdata` array, a member of this class of object.

`~datavalues_itemcp()`
Destructor for object, should be called when object is destroyed to ensure memory is properly freed.

`void setActualData(datavalues_itemcp*)`
Used to copy data from temporary `currentitem` to new item in memory. Copies from array member of current item to new item.

`void createActualData(unsigned char*)`
Directly copies data from `Image workingImage` object into new datavalues_itemcp object. Performs the binning according to binning parameter set in `params.txt` file. Receives character pointer array and stores each calculated value in the object array.

`unsigned short* getActualData()`
Gives pointer to first item of data array of shorts for an object.

`void writeOutDirect(int)`
This is not an essential function and was mainly created for testing. It writes out a file of the image array as a single line of numbers directly from the `currentitem` object to file without storing dynamically in memory. This may be marginally faster than storing in memory then writing to disk by a couple of milliseconds. It also performs binning prior to saving. It numbers the file according to the provided number `iloop`.

5.3.2 Data list class

`class Actual_data_listcp`
This class participates in the dynamic memory allocation process for storing pictures in memory. An image is created as a datavalues_itemcp, and then stored dynamically as a member of the data list. The data list contains members including, importantly, pointers to the first and last items in the list, `datavalues_itemcp* firstcp, lastcp`. These work in conjunction with the list item `datavalues_itemcp* nextcp` to coordinate the linked list structure for dynamica memory allocation, such that each item in the list points to the next item, and the list itself knows its first and last items. These are very important for the proper functioning of the dynamic memory allocation and should be edited with great care. The list object also knows the number of items it contains, the number of pixels in each array for each object, and the max number of images it can have.

`Actual_data_listcp(const Actual_data_listcp& Add):maxnumitemscp(0)`
Copy constructor for copying a data list. This is `private` to ensure that the user does not attempt to copy the list using this as this would mess up the structure and not ensure all the variables are initialized properly.

`Actual_data_listcp(unsigned int,unsigned short)`
Constructor for the data list. It must be passed the number of entries in the arrays for its item objects, as well as the maximum number of items it can hold. The pointers to the first and last items in the list, `firstcp` and `lastcp` are initialized to null, and number of items in list initialized to 0.

`~Actual_data_listcp()`
Destructor for the data list. This goes down the linked list using the pointers to items `firstcp, lastcp, nextcp` and deletes each item, freeing up the memory each item is taking, properly, to ensure there are no memory leaks.

`int getNumDatavaluesItems()`
Returns number of items in the list.

`bool addDatavaluesItem(datavalues_itemcp* currentitemAdd)`
Adds a datavalues_itemcp object to the data list by copying the info from the temporary current item. It performs operations necessary to add it to the linked list structure and ensure all the member variables and pointers are set correctly in the new list item and the list itself.

`bool removeDatavaluesItem(datavalues_itemcp* currentitemAdd)`
Frees memory of the first item in the list and deletes the array of data, ensuring that the memory is freed properly and there are no memory leaks. This ensures the proper reassignment of pointers.

`datavalues_itemcp* getFirst()`
Returns a pointer to the first item in the data list.

`int getDatavalues_sizecp()`
Returns the number of elements in the array for each item in the data list.

5.3.3 File writing class

`class FILE_item_listcp`
Contains all structures and functions necessary to write out data from datavalues_itemcp to

file as an array or matrix. This class takes `datavalues_itemcp* currentDataItem` in the list pointed to by `Actual_data_listcp* pdataList` and writes it to file.

`FILE_item_listcp(Actual_data_listcp* pdatalist)`
Constructor for the file item list, all it needs is the address, a pointer to, a data list item `Actual_data_listcp`, to which it initializes its own member variable pointer to a data list.

`~FILE_item_listcp()`
Destructor for the file item list.

`unsigned short getNumFileItems()`
Returns number of file items for the file item list, a number which increases every time another file is written.

`bool write_FILE_itemcp()`
Writes array in `datavalues_itemcp`, the first item in the data list, to file. It numbers the file according to its place in the list.

`bool write_FILE_itemcp(int k_width)`
Writes array in `datavalues_itemcp`, the first item in the data list, to file as a matrix with width `k_width`. It numbers the file according to its place in the list.

`bool write_FILE_itemcp(int k_width,int filenum)`
Writes array in `datavalues_itemcp`, the first item in the data list, to file as a matrix with width `k_width`. It numbers the file according to the number provided as `filenum`.

5.3.4 Other

`void writeDirectFromImage(unsigned char* datavalues,int addBin, int width, int height, int iloop)`
Writes array directly from a pointer to data in `Image workingImage` to file. This is intended mostly for testing purposes and not essential. It may be marginally faster than going through the dynamic memory allocation process. It performs binning as well. It numbers the file according to the number `iloop` passed to the function.

6 Physica Scripts on trinatlt2

`trinat@trinatlt2:/home/trinat/flycapture/flycapcode/src/cpAsyncTriggerEx/`

6.1 Initiate Physica Defaults

`physica_defaults.pcm`
Initiates defaults for Physica.

6.2 Put in Initial Values

`physica_initvalues.pcm`
Initializes values including file name prefixes and suffixes, index of image.

6.3 Read In Background Picture

`physica_bgpic.pcm`
Reads in background image `cp0.out`, plots it as a density plot, along with the projections, and creates projections that it can use later for background subtraction.

6.4 Read In and Display Picture

`physica_displaypics.pcm`
Reads in respective image according to value of index `ind`, displays, and displays background-subtracted projections, as well as overall intensity sum over past 20 or so images.

7 Programs to Set/Test DAC's and GPIO Pins

`pi@trinatrpi2:/home/pi/RunTrapSequence_cp`
Briefly, here is a list of programs that can be used to test the DAC's and GPIO Pins.

`setDACoutputs(.cpp)`
This allows you to set any of the DAC's to a test value in a loop. Must be executed with sudo.

`testAOMAttenLevel(.cpp)`
Allows you to set the level of the AOM Crystal Tech in a loop. Must be executed with sudo.

`testGPIOs(.cpp)`
Unfortunately I did not have time to develop a simple program to test each individual GPIO pin in a loop, though this would be fairly easy. This program requires you to manually edit which pins go on and off, but you can test them that way and recompile.

`testInputLevel(.cpp)`
This was used to test the input level of the Pi by hooking up a voltage from the LabJack board to an input pin. If using this again, you must be very careful not to cause over-current or voltage damage.

`testCAM(.cpp)`
This was created to test the trigger and strobe capabilities of the camera.

8 Pi Backup and SD Card Cloning

8.1 Backing Up to SD Cards

`pi@trinatrpi2:/home/pi/BackupLog/`
Backups of the Pi are set up as `crontab` jobs. The document explaining this is in the directory indicated, but will be briefly reiterated below.
We have two scripts that are set to back up to each of the SD cards. Each is recognized by a unique UUID that is different for each storage device and if different SD cards are ever used, this value will have to be found. Specific instructions are in the shell scripts for finding this. The full backup command in the shell script for the main ext4 partition is `sudo rsync -av --delete-during --exclude-from=/home/pi/BackupLog/rsync-exclude.txt --log-file=/home/pi/BackupLog/rsync/$(date +%Y%m%d)_rsync_main.log / /mnt/rpibackup1/_`
The other command for backing up the boot partition is very similar. `sudo` is necessary to preserve root ownership. `--delete-during` ensures that the correct files are deleted to ensure

proper synchronization, `--exclude-from` tells us which files to not copy, since some only exist during system run time and are not needed to boot and run the pi, though the files still must exist, `--log-file` keeps a log of the backup.
To add the task to the `crontab` such that we can have it backup every day at a specified time, we type `crontab -e`. It instructs you to enter your command as
`m h dom mon dow command`
this means 'minute hour day month day-of-week command'. So if we enter
`30 01 * * * /home/pi/BACKUPSCRIPTNAME.sh`
into the crontab editing file, it will backup our script every day at 1:30am.
They use `sudo rsync -av` to back up to each drive. the `sudo` is necessary to preserve root ownership of necessary files - without this, the resulting system will not boot.

8.2 Cloning SD Card Using `rsync`

`trinat@trinatlt2:/home/trinat/RPi/RPi2_CompleteBackupFiles/RPi2_B6_clonebackup_june16_2015`
`trinat@trinatlt3:/home/trinat/RPi/RPi2_CompleteBackupFiles/RPi2_B6_clonebackup_june16_2015`
Files are located in the above directories that can be used to create a bootable identical real-time system with the necessary libraries and the majority of the programs necessary to run the trap on the Pi, though not the most updated versions, as this was cloned in June 2015. There are text files in these directories instructing the user what to do to backup or clone the Pi, but it will be reiterated below.
The rsync method is preferred for cloning the card from an external laptop, as it is deemed safer than the `dd` method in the next section.
Assuming you have system files in `_boot` and `_` folders for which root ownership has been preserved for the relevant files (true for the files in the listed directories), insert an SD card into the card reader (you will need an adapter for the micro-SD cards). Use gparted to partition the new card, in sequence with 4MB unallocated space, a 56MB FAT16 partition (labelled /boot), then the rest of the space make an ext4 partition (labelled /). Use `sudo rsync -av` to transfer files respectively to each partition from each of the respective backup folders. the `sudo` is necessary to preserve the root ownership of necessary files - the system won't boot if this is not done properly.
The most updated programs for the trap can be copied to the user directory `pi` from the latest backup, or failing that, from `trinat@trcomp:/home/cpreston/RPi/RPi2_Backup_B2/...` picking the most recent backup.

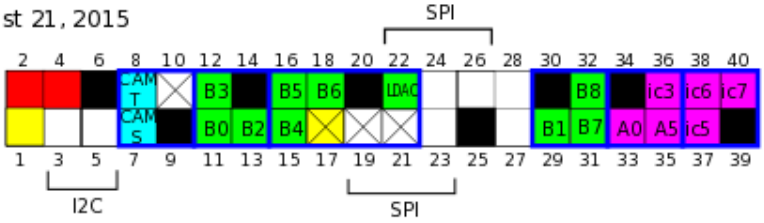
8.3 Cloning SD Card using entire Image

`trinat@trinatlt3:/home/trinat/RPi/RPi_Images/emlid-raspberrypi2-raspbian-rt-20150327.img`
`trinat@trinatlt2:/home/trinat/RPi/RPi_Images/emlid-raspberrypi2-raspbian-rt-20150327.img`
This is the less preferable method, but also works, for cloning an SD card image. The files indicated above are both real-time Raspbian images from Emlid. Note that these do not contain the bcm2835 library or the labjack library, and so these would have to be reinstalled from scratch, so do not use this method if you can do the previous one.
If the system and all the backups really have failed miserably, then proveed as follows. Using the `sudo dd` command with `bs=4M`, write the entire image to the new card. Make sure you look up the `dd` command as this is dangerous and you can write over your entire hard drive if you use this unwisely. If the target card is smaller than the original card used to write the image, this will not work as `dd` will run out of space.

9 Pin and DAC Diagram Reference

Raspberry Pi 2 Model B GPIO Pin Diagram: DC/ACMOT Trap Control Sequence Testing

Claire Preston
August 21, 2015



GPIO Pins

CAM T	CAM_TRIGGER
CAM S	CAM_STROBE
A0	ACMOT ON/OFF
A5	PUSHBEAM ON/OFF
B0	SIGMAPM (ON=PLUS)
B1	OPCYCLE ON/OFF
B2	TRAP ON/OFF
B3	SIGMAPM_TO_NUCLEAR (ON=PLUS)
B4-B7	DAC_BITS 0-3
ic3	TO_NUCLEAR ON/OFF
ic5-7	AOMPOWERLEVEL noatten-fullatten
LDAC	ABDAC transmission

DACs

ABDAC0	DAC7 - cooling
ABDAC1	DAC2 - power repumper
LJDAC0	DAC6 - step to nuclear
LJDAC1	
LJTDAC4	DAC4 - AOMs power
LJTDAC5	DAC9 - freq diode laser
LJTDAC6	DAC8 - step
LJTDAC7	DAC3 - B Field

TE Connectivity 2x2
AMP Connectors 4-87456-9
5-87456-0

Stores connector
4x2 3-4/1188
87456-3

LEMO
(To Nuclear Acquisition)

TE Connectivity 487526-3
(To Precision Amp)

BNC
(To Camera)

Shared Ground

3.3V

5V