Notes on Operation of Spark system

**Logging onto the Clean room Computer**

QC-PC is a local admin account on the computer, as opposed to the typical Fermilab domain accounts

When logging on, ensure that you are logging onto the “Local” instead of “FERMI” domain

If it says fermi, add “PPD-138201/” to the beginning of the username – this could change based on hardware – there’s a tooltip listed under ‘additional domains’ or similar that tells you what to use

Username: QC-PC

Password: Mu2eqc2022!

**Starting the spark monitor script**

After logging in, make sure you start the autohotkey script titled "Run this to prevent logoff" to prevent the computer from locking after 15 minutes of inactivity.

This script just wiggles the mouse every minute, you can tell it’s active if there is a green H icon in the taskbar

Files are stored in the ‘root directory’ determined by the DroegeMonitorv2.py script. By default, they are stored at (C:\Users\QC-PC\Desktop\Current Monitor\)

Check to see if the Pi Pico is correctly set up

Pico kludge board should have 3 lemo inputs, each connected to the Droege current monitor port for each table.

Pico should also be connected via USB to the computer

Troubleshooting Pico

If Pico is not detected, unplug it and plug it back in – this fixes it 90% of the time

If still not detected, check to make sure the COM port has not changed

Open “Device manager” on windows (via search bar) and open the tab for Ports (COM & LPT) or similar

Edit the DroegeMonitorv2.py script to the correct port

If all else fails, ask Vadim or reflash the pico board with the main.c code provided

Start the DroegeMonitorv2.py script

Script can be started via command line or through compiler

Via command, cd to the root directory then type “py DroegeMonitorv2.py”

Script should run normally and begin collecting data

Exit the code by switching to command line and pressing Ctrl+c to ensure data is saved

Accessing the data

Files are all stored under the root directory under the assigned timestamp folders

Hitrate files store the hitrates of each table independently by hour

Current files store the current measured by all three tables in one csv file with timestamps

**Start the VidCapture Script**

Vidcapture script can be started by doubleclicking the correct .exe file

These files are precompiled and should be placed in the root directory

VidCapture.exe corresponds to the single camera system

VidCap6cam.exe corresponds to the 6-camera system

Bgsubtract.exe collects the single frame images used to make heatmaps

A console output should pop up letting you know if the script has started successfully

**Creating the ‘HeatMaps’**

Images for each event should be present in the corresponding folders

This is something that should eventually be automated

Before running the script, a single image should be taken with the lights turned on for the overlay

Save this image in the corresponding folder as baseline.jpg

Move a copy of Overlay.py to the folder with the corresponding date you’d like the heatmap for

Edit the Overlay.py line 6 to be the filename of one of the .bmp images in the folder

Run the script and view the heatmap, saved as overlay.jpg

**Editing the Scripts**

Scripts can be edited by opening Visual Studio on the Desktop via search

Projects are listed in the Welcome Section, select the project you’d like to edit

I’ve preloaded all three projects onto the clean room PC for you to make changes

Make any changes to the code via the editor

Press the Green arrow (Local Windows Debugger) to build and run the script

Alternatively, you can hover over the Build tab and press “build solution” to build without running

Building the project will generate a (project\_name).exe file which can be placed and ran anywhere (It’s best to move this file to the root directory defined earlier before running)

The .exe file is within the visual studio ‘repos’ folder, (C:/Users/QC-PC/Source/Repos/(project\_name)/x64/Debug/(project\_name).exe)

**Creating a new OpenCV Project from scratch**

This is where things get a bit complicated

Open Visual Studio

Create new project

Select Console Application and name the project

Under the Project Tab, select (project\_name) properties (with the small wrench)

Under the VC++ Directories tab, edit the following entries

Include –

C:/opencv/opencv/build/include

Library Directories –

C:/opencv/opencv/build/x64/vc15/lib

Under the C/C++ tab edit the following entries

Preprocessor - Preprocessor definitions -

\_CRT\_SECURE\_NO\_WARNINGS

Under the Linker tab edit the following entries

Input – Additional Dependencies

Opencv\_world455d.lib

Opencv\_world455.lib

Test the code by adding #include <opencv2/opencv.hpp> and running

Alternatively, I’ve created a project template with all of the above included and working

When creating a new project select OpenCV project instead of console application and begin coding

This won’t transfer over to other PC’s and is a bit more technical but if you can figure it out it’s way easier

**Building OpenCV on Windows**

Building OpenCV is a pain, I’ve already installed it on the clean room PC for you to use (Under C:/opencv/opencv)

If we get a different computer to run the scripts on, you’ll need to build opencv via cmake on the new system before you can build and run code.

Directions on how to do so are linked

<https://cv-tricks.com/how-to/installation-of-opencv-4-1-0-in-windows-10-from-source/>