Practical-Focus Guide for the calibration system

1.Gantry system

To control the gantry system, we use Qt and Arduino program. We can take a look of the code in Arduino first:

The gantry system is actuated by four stepper motors and we use FMDD50D40NOM model motor driver to control those motors. Each motor is controlled by `DirPin` and `StepPin`. `DirPin` determines the direction of rotation of the motor and `StepPin` actuates the motor by receiving phase signal.

We also define two states for the gantry system, while state = 0 means the system is working properly and state = 1 indicates that the gantry stage is moving out of bound so the system will stop immediately. The change of state can be trigger by the proximity sensors, there are 6 sensors installed on both ends of each leadscrew. If the gantry stage is too close to the sensor, the switch inside the sensor will changes the output voltage from 24 V to 0 V. Because Arduino boards can't read 24 V input, optoisolators are also used in the circuit to convert the voltage output from 24 V to 5 V. (The detailed circuit of the proximity sensor will be shown in Figure 1 and Figure 2)

```
void endstop_start_x(){
  Serial.println("stop_start_x");
  state = 1;
}
```

We use interrupt pins on Arduino Mega board to receive the signal from the endstops and change the state from 0 to 1.

For the

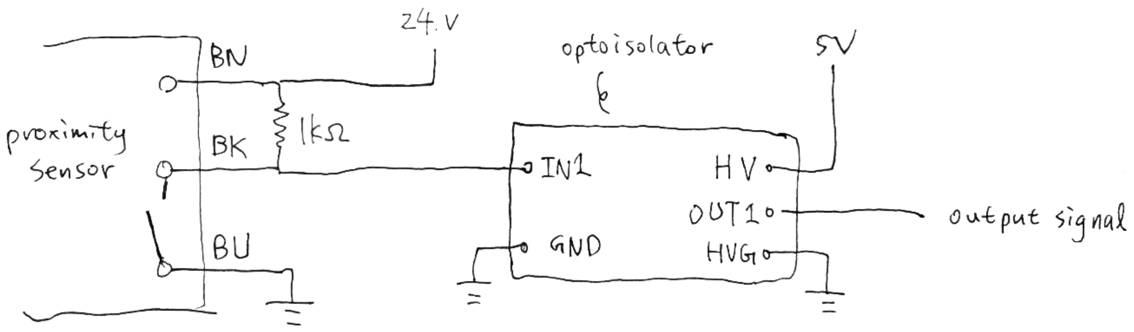

Figure1: The circuit of proximity sensor and optoisolator

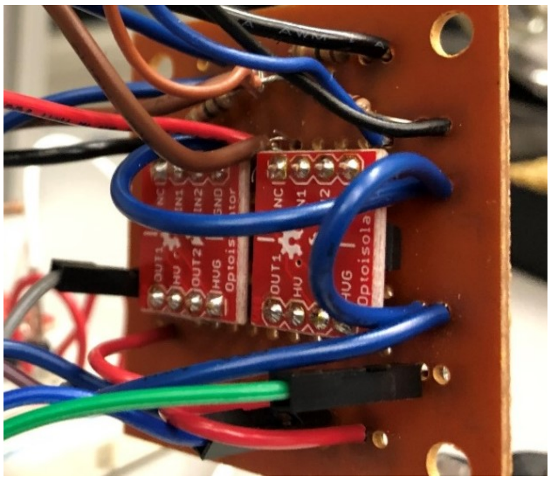
Figure2: The soldered circuit of the optoisolators

The current circuit board is not properly mounted or fixed, so some extra works may need to manufacture a plane or box to secure the circuit board.

There are 6 proximity sensors for the endstops but one of them didn't work. Therefore, the moving range of the gantry stage still need to be set up in the program to prevent collision. In addition, it's better to have the information of the position of the probe so that we can better integrate the system. The boundary is defined in Qt gamepadmonitor.cpp file as the following:

```
int x_desired = 0;
int y_desired = 0;
int z_desired = 0;

int x_max = 20000;
int y_max = 100000;
int z_max = 100000;
```

x_desired, y_desired, and z_desired are the variables to record the current position of the gaussmeter probe. x_max, y_max, and z_max define the maximum position value that can be reached for each axis. x_min, y_min, and z_min can also be added in the future to better fit the moving constraints in the workspace of the magnetic actuation system. For example, currently the setting of the boundary doesn't consider the gaussmeter probe, which means the gantry stage itself is safe to move around the workspace without hitting any object. However, when the probe is mounted on the gantry stage, the probe might hit the actuation system due to its long shape. Therefore, the boundary should be reset according to the actuation system and the probe.

The XBOX gamepad is used to control the gantry system, and there are some frequently used commands we are using now:

Press "B", the probe will go to the zeroing point of the gantry system.

Press "L2" the probe will go to the origin of the actuation system

Press "Y" the gantry system will start the scanning routine

Notice that the XBOX gamepad only worked on Windows when plugged-in and configured as Player-1, so be careful not to touch the “Xbox” button (The green one in the middle) because that will change the user configuration to Player-2.

2.Gaussmeter

The upper and lower measurement bound of the magnetic magnitude is set to +300 mT and -300 mT, respectively to suit the current magnetic actuation system. Be sure to calibrate the gaussmeter probe using the zero gauss chamber before using. The gaussmeter has BNC terminals at the back of the unit; it converts the magnetic magnitude value into an analog voltage signal and then send out to the DAQ system.

The gaussmeter probe is currently mounted on the gantry system, although the gantry system can perform accurate motion control in X, Y and Z axis. However, the 3 axes of the probe are not perfectly aligned with the coordinate of the calibration system, which might need extra works to further calibrate the axis distortion error.

3. DAQ

For the DAQ system, there is a useful library in C++ environment: "NIDAQmx.h". The setting of the library’s function is in the *daq.cpp* file, including the input pins, the terminal configuration, the sampling rate, etc. The current setting are ready to sample the voltage data from the gaussmeter, but more information can be found in the links below:

Programming tutorial: http://www.ni.com/tutorial/5409/en/

DAQ pinouts: https://www.ni.com/documentation/en/multifunction-io-device/latest/usb-6210/pinout/

The Qt program samples the voltage data through the DAQ and combine this data with the position data of the probe and the rotation data of the magnets, then write the information of this data into a txt file (the path of file can be set in the daq.cpp).

4. Experiment Procedures

The calibration experiment procedures can be break down to the following steps:

1. Use zero gauss chamber to zero the calibrate the gaussmeter probe (The probe hasn’t been mounted on the gantry stage yet)

2. Press “B” button on the XBOX gamepad, the gantry stage will move to the zeroing point of the gantry system

3. Press “L2” button on the XBOX gamepad, the gantry stage will move to the origin of the magnetic actuation system.

4. Mount the gaussmeter on the 3D-printed mount on the gantry stage.

5. Press “Y” button on the XBOX gamepad, the system will start the

calibration process.

6. When the calibration process is done, remove the gaussmeter probe from the gantry stage.

7. Press “B” button on the XBOX gamepad to move the gantry stage back to the zeroing point of the gantry system.

8. Open the txt file written by the Qt program in MATLAB and run the interior-point algorithm program to obtain the calibrated parameters.

For the current experiment procedures, step 4 and step 6 may cause undesired error when we remove and mount the gaussmeter probe every single time. The reason that we have step 4 and step 6 now is that when the probe is attached to the gantry stage, the moving range of the gantry stage is limited and therefore couldn’t reach the zeroing point of the gantry system (The zeroing point is determined by the endstops). But we have to zero the gantry stage every time when we want to do an experiment. A possible solution is to move the endstops closer to the center so that the zeroing point can be redefined within the boundary of the actuation workspace.

5. MATLAB code

The MATLAB code for the calibration program can be found in the link below:

https://github.com/NelsonLee-NTU/MIE498-MATLAB.git

The code includes magnetic field simulation, least-squares calculation, and the program of interior-point algorithm, etc.