Fall | 14

Weekly Status Report

Wesley Myers

November 18th – 24th, 2014

wesley.y.myers@gmail.com

Table of Contents

Executive Summary

The main objective for this week was the get core system functionality working in terms of targeting.

Website Development

- MySQL database added
- Additional Serial Needs

Website Development

Part of the core concepts of this project is scanning over a target list. What we want to do here is let the user set a series of targets and then have the system scan over them. In order to do this, a database needs to be chosen. MySQL is the database of choice due to its common use. As there is a lot of code in development, next week's report will contain most of the new code.

Installing MySQL is fairly straightforward. Just execute the following.

```
sudo apt-get update
sudo apt-get install mysql-server
```

In order for MySQL to run more efficiently on the Raspberry Pi, we need to edit the configurations a bit from the standard distribution.

```
sudo mv /etc/mysql/my.cnf /etc/mysql/my.cnf.bk
sudo cp /usr/share/doc/mysql-server-5.5/examples/my-
small.cnf /etc/mysql/my.cnf
```

Now in my.cnf, change the file to reflect

```
query_cache_size = 8M
```

And then restart mysql

```
sudo service mysql restart
```

Since most of the work already done uses python, I'll be using python scripts to interact with the database. Apparently python only comes with mysql drivers for SQLite, so thus drivers must be installed for python to interface with MySQL.

```
sudo apt-get install python-mysqldb
```

And now you should be set!

CREATE TABLE

The first thing that needs to be done is create a new database. Change directories to /var/www. Log into the system by executing the following.

```
mysql --user=”root” --password=”system”
```

You should now be inside of MySQL and should have a command prompt. Type in the following command

```
CREATE DATABASE nslaser_db;
```

Woohoo! You should now have a new database to work with. Exit out of MySQL by typing “quit” into the prompt. Now we need to add tables. So go ahead and log back into MySQL, but include the database name.

```
mysql --user=”root” --password=”system” --database="nslaser_db"
```

Now let’s create 2 tables for the system to interface with. The first is a target list. This will store our target information, including the name. The second is the system information. This will enable us to keep track of the system’s pan-tilt position, plus the last laser read.

```
CREATE TABLE targets (name varchar(30), pan int, tilt int, distance float);
CREATE TABLE system_info (pan int, tilt int, distance float);
```

Adding values into the tables is fairly straightforward.

```
INSERT INTO targets (name, pan, tilt, distance) VALUES ('copper', 90, 90, 1.23);
INSERT INTO system_info (pan, tilt, distance) VALUES(90, 90, 0.0); //default settings
```

The user may want to erase the target list.

```
DELETE FROM targets;
```

In the case of the system_info table, we just want to update values.

```
UPDATE position SET pan = 100;
```

In case you want to delete a table from the database, then just do the following.

```
DROP TABLE table_name
```

The python serial script was modified to update the system_info table whenever a reading was made, or a position was updated. This allows easy addition of targets with the last known move.

In the script, it is important to include the following. This will enable you to interact with the database.

```
import MySQLdb
db = MySQLdb.connect(host="localhost", user="root", passwd="system", db="nslaser_db")
cur = db.cursor()
```

With cur (cursor) you can make calls to the database. Instead of typing in the command on the MySQL prompt, you'd put it here.

```
cur.execute("SELECT * FROM system_info")
```

In the python scripts, it is important to make sure you close out the database.

```
db.commit()
cur.close()
db.close()
```

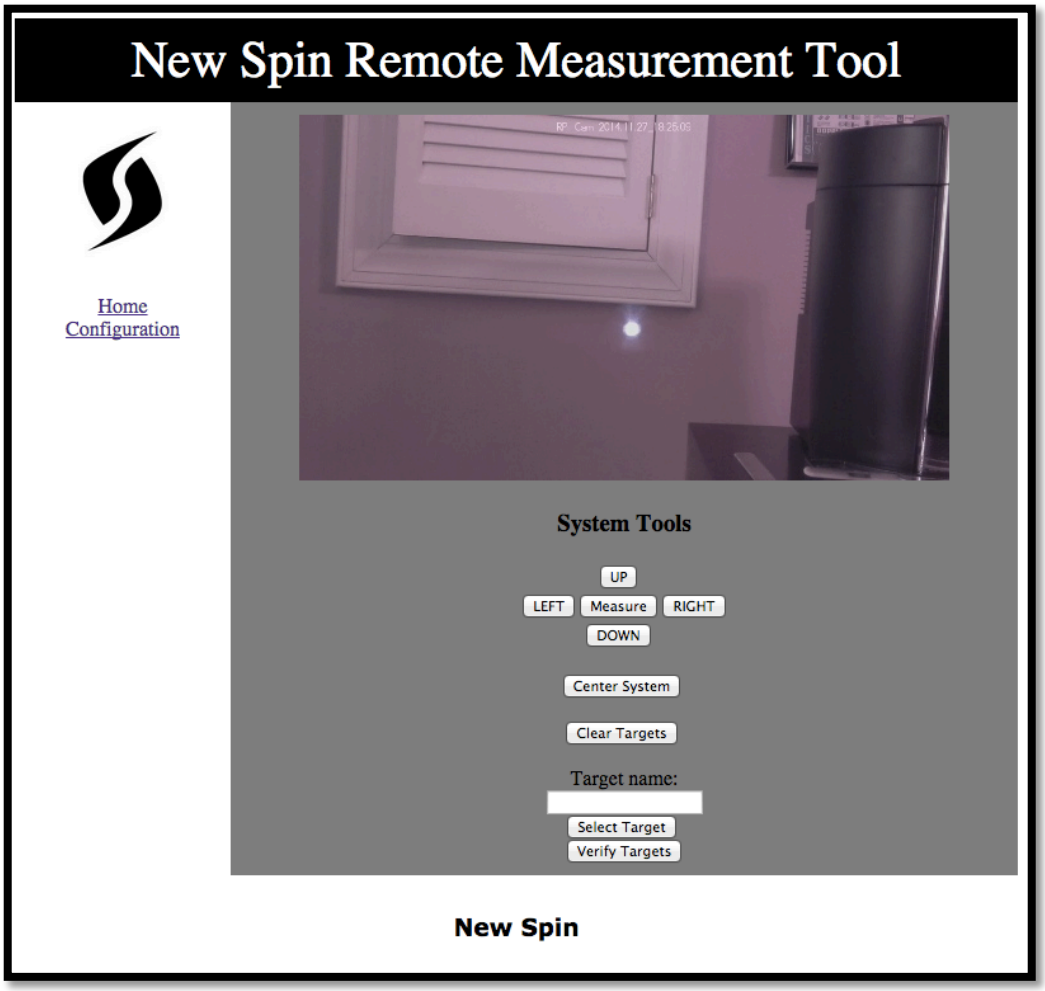

Figure 1 - Current Website Layout