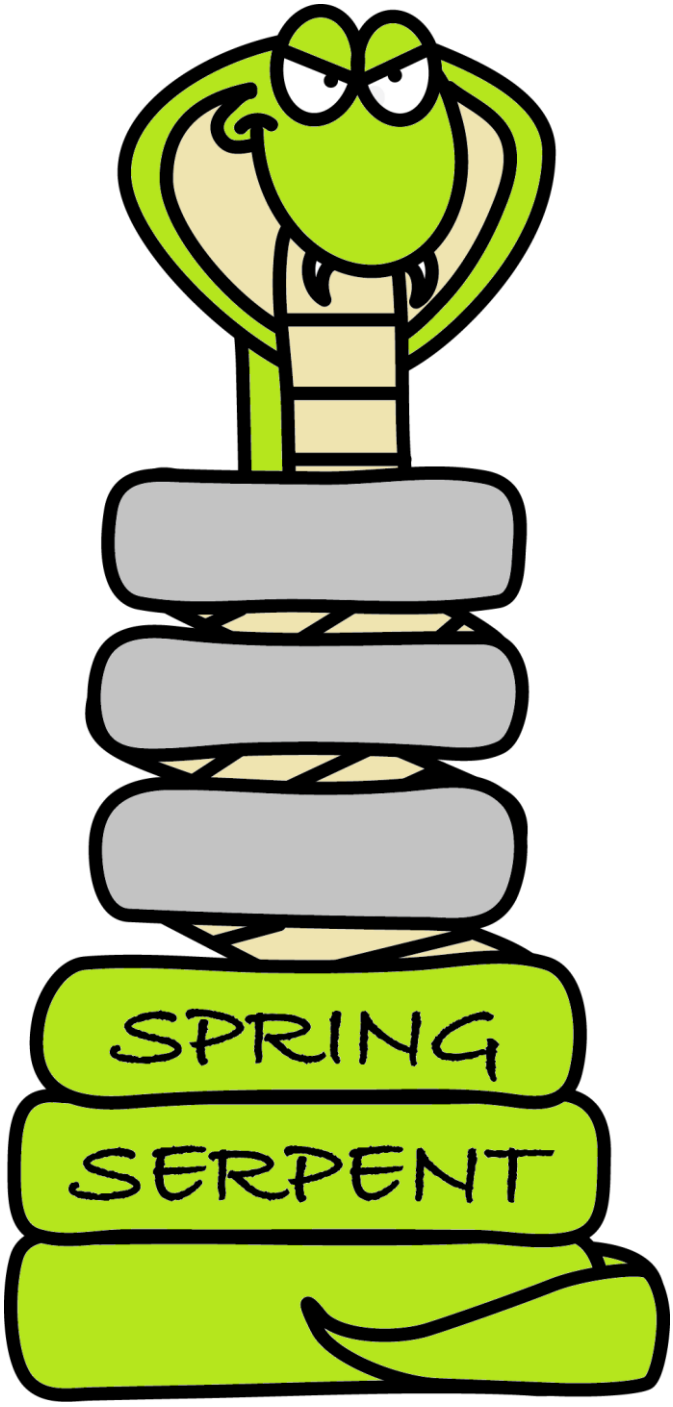
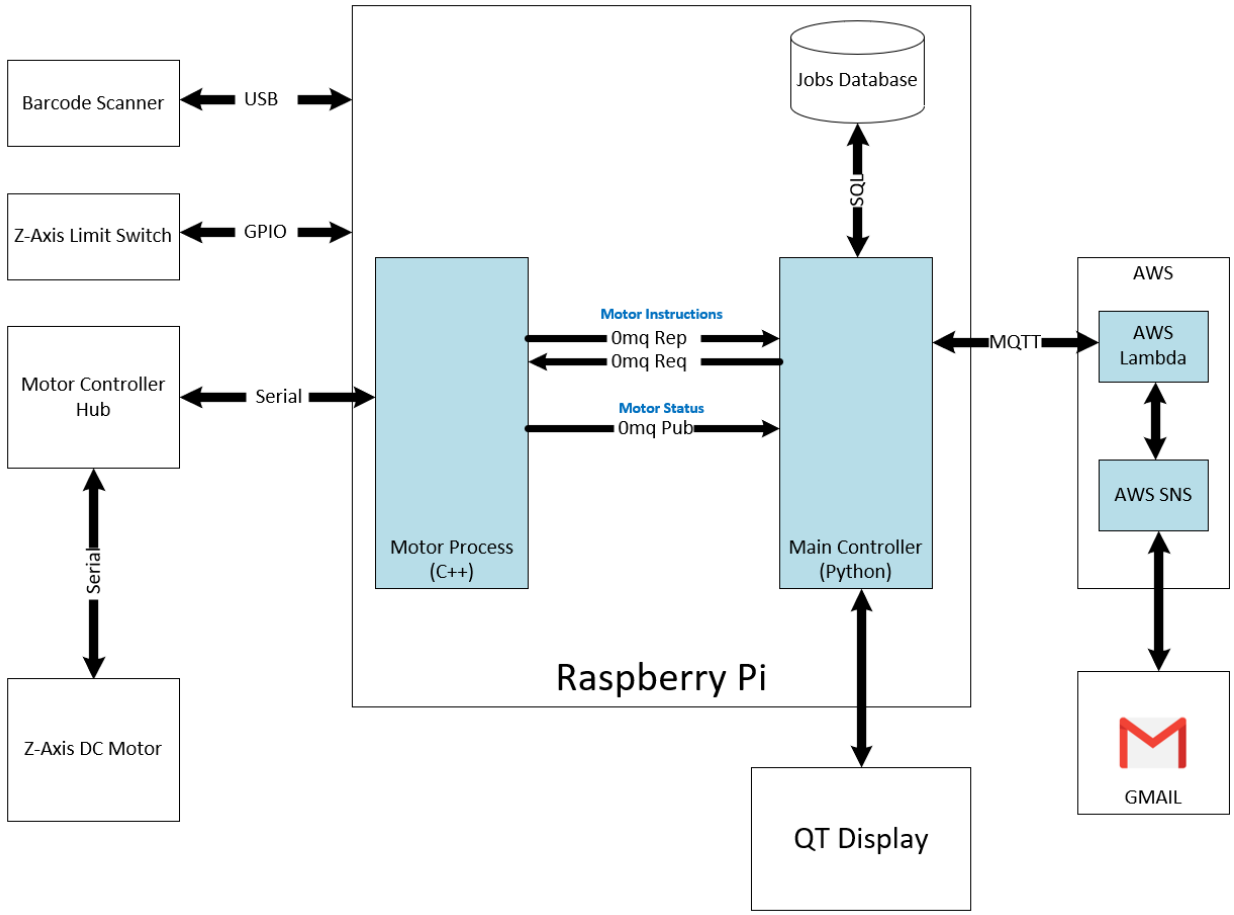



# An Automated Spring Winder

Created by Glenn Feinberg

# Table of Contents

## System Architecture




The Spring Serpent is a raspberry-pi controlled motor setup that automates the process of spring creation. To demonstrate the prototype, the z-axis motor was used to demonstrate the functionality and feasibility of the motor controlling and user interface design.

The raspberry pi is the heart of the system that is responsible for all of the control and decisions of the product. It is setup with two main processes, the main controller which is responsible for:

- Graphical User Interface (QT) to display information and accept user input
- Creating motor instructions
- Communicate/Connect with amazon webservices
- Interface with the SQL Database

The other process is the motor controller process which is responsible for the communication of the motor controller hub via serial which controls and interfaces with the z-axis motor also via serial. The two of these processes communicate with varying 0mq messaging types since they're run on separate processes. The motor status from the motor process is via a pub/sub method which has the motor process as the publisher and the main controller as the subscriber. The motor instructions are done with a request/reply structure where the motor process is the server and the main controller is the client.

The main controller will request an action and motor controller will respond on success of the receipt of the message.

The main controller contains the QT code that interacts with the touchscreen and displays the graphical user interface while also providing input methods to perform varying tasks. Additionally, the controller communicates with AWS via MQTT. After connecting to the AWS Thing, the MQTT is processed via a lambda function which then directs the data to an SNS service that sends an email indicating job status.

Some additional features incorporated are a z-axis limit switch to help with the homing process and a barcode scanner used to assist in loading previously created spring jobs. The spring jobs are stored in a SQL database on the raspberry pi and the main controller interfaces with this via SQL using SQL Alchemy as ORM.

## Project Deviation

The project did not change much from the original scope of work, except with the removal of the AC motor. A single motor was enough to demonstrate most of the tasks and sourcing a large AC supply proved difficult. The architecture stayed the same throughout the project, but the user interface changed after paper prototypes with the user. Some major changes to the UI:

- Added a home-screen to allow for an easier method of going to screens
- UI's do not have a navigational bar, but instead provided proceed and back buttons for simpler navigation
- The job library screen was given it's own dedicated screen
- Some screens that are not functional are the info, warnings, and settings. The icons are present, but no screen created at this time

## Resources

### General

- https://htmlcolorcodes.com/color-chart/
- https://unix.stackexchange.com/questions/106480/how-to-copy-files-from-one-machine-to-another-using-ssh
- https://www.linode.com/docs/tools-reference/tools/find-files-in-linux-using-the-command-line/
- https://stackoverflow.com/questions/18900855/how-to-work-with-external-libraries-when-cross-compiling
- https://medium.com/@jcrbcn/installing-the-exar-usb-driver-on-the-raspberrypi-for-teknic-sc-hub-39de533f0502
- https://stackoverflow.com/questions/1896369/how-to-use-a-class-object-in-c-as-a-function-parameter
- https://linux.tips/programming/how-to-install-and-use-json-cpp-library-on-ubuntu-linux-os

### 0mq

- http://zguide.zeromq.org/cpp:hwclient
- http://zguide.zeromq.org/cpp:hwserver
- https://stackoverflow.com/questions/48414559/pyzmq-req-rep-multiple-clients-zmqerror-with-polling
- https://stackoverflow.com/questions/37765847/send-json-object-through-zmq-with-c-client-python-server
- https://stackoverflow.com/questions/54045421/not-able-to-parse-jsoncpp-object-received-in-0mq-socket
-

### PyQT

- https://kushaldas.in/posts/pyqt5-thread-example.html

### SQL

- https://docs.sqlalchemy.org/en/13/orm/tutorial.html
- https://towardsdatascience.com/sqlalchemy-python-tutorial-79a577141a91
- http://zetcode.com/db/sqlalchemy/

### AWS

- https://docs.aws.amazon.com/iot/latest/developerguide/iot-lambda-rule.html
- https://docs.aws.amazon.com/lambda/latest/dg/with-sns-example.html
- https://aws.amazon.com/premiumsupport/knowledge-center/access-denied-lambda-s3-bucket/
- https://docs.aws.amazon.com/lambda/latest/dg/with-sqs.htmlhttps://boto3.amazonaws.com/v1/documentation/api/latest/guide/sqs.html
- https://docs.aws.amazon.com/code-samples/latest/catalog/python-sqs-receive_message.py.html
- https://startupnextdoor.com/adding-to-sqs-queue-using-aws-lambda-and-a-serverless-api-endpoint/
- https://boto3.amazonaws.com/v1/documentation/api/latest/guide/sqs-example-sending-receiving-msgs.html
- https://docs.aws.amazon.com/iot/latest/developerguide/iot-moisture-raspi-setup.html

## Project Observations

1. The C++ aspect of the project proved to be slightly harder then expected because of the cross-compiling process with the sFoundation library for the vender motors. After setting up the development environment (makefiles, import statements, cross-compiling) things went much smoother
2. Getting devices to connect with Linux can sometimes prove challenging. Both the touchscreen and barcode scanner did not easily connect with the raspberry pi and countless hours were spent just getting the aspects to work

3. Developing a UI that makes sense to both an engineer and a user is a very interesting experience. As someone who uses applications on the daily, a complex maneuver method sometimes comes second nature and designing for a different user can sometimes prove difficult when you think it makes sense to you. Trying to overdesign ideas seemed to happen a lot when the method of K.I.S.S. (Keep it simple stupid) should of occurred.
4. There are a lot of methods to communicate with devices and sometimes it is better to just dive in and see what may work then overthinking one method or trying to force one method to work when it may not be the right answer. I learned the idea that you should make one just to throw it away to get some ideas out there.