ABM Interaction Setup

Release 1.0

Feb 19, 2020

Contents

Contents:

CHAPTER 1

Setup

1.1 What you'll need

- A monitor
- A mouse and keyboard
- Wireless internet
- An Amazon Web Services account
- An account with Fitbit
- A tablet for the user interface
- A Google account to get an app for the tablet

QT has two ports: one USB-C and one USB-A. You'll need to use the USB-C port for display and figure out how to control a mouse and keyboard. I suggest a USB-C hub that has a display port that you can use (USB-C or HDMI, for example) and has USB-A ports for your mouse and keyboard. Alternatively, you can use a USB-A hub to connect your mouse and keyboard to QT.

Note: If you have trouble using your mouse or keyboard through a USB-C port, try flipping the USB-C input going into QT. In theory, USB-C should go both ways, but in practice, sometimes not.

1.2 Basics

1.2.1 Turning QT on and off

To turn on QT, just plug in power to QT and it will boot.

To turn off QT, there are two options:

1. Press the button on the backside of QT, near its feet.
2. Login to the head computer (see below) and do `sudo shutdown`.

If you do `sudo shutdown` on the body computer, you only turn off the body computer—the head computer is still on.

Note: If you unplug QT to restart QT, it may mess up the boot timing of the two computers. Probably one of them takes longer because it boots in recovery mode. This screws up how the head and body computer network. You will not be able to connect to the head computer from the body computer. If this occurs, simple restart QT by pushing the button on its backside.

1.2.2 Accessing QT's body computer

When you connect a monitor to QT and turn QT on, you will start on QT's body computer.

1.2.3 Accessing QT's head computer

To setup the head, you must Secure-SHell into it (SSH) from QT's body computer. To do this

0. Turn on QT.
1. Open a terminal.
2. Type the following and hit return:

```
ssh qtrobot@192.168.100.1
```

1.3 Head

1.3.1 Turning off the default face

0. Secure-Shell (SSH) into QT's head computer:

```
ssh qtrobot@192.168.100.1
```

1. Update QT:

```
cd ~/robot/packages/deb
git pull
sudo dpkg -i ros-kinetic-qt-robot-interface_1.1.8-0xenial_armhf.deb
```

Note: If the `git pull` step fails, the head computer might be having trouble with it its network. You can check this with `ping google.com`. If there's nothing, there is a problem with the network. To fix this, the best think we've found is to restart QT: `sudo reboot`.

2. Edit a configuration file to turn off QT's default face:

   a. Open the configuration file:

```
sudo nano /opt/ros/kinetic/share/qt_robot_interface/config/qtrobot-interface.
↪yaml
```

b. Change the line that says `disable_interface: false` to `disable_interface: true`

Note: You can reboot to see these changes take effect, or continue on and we'll reboot eventually.

1.3.2 Setting up our code

0. Secure-Shell (SSH) into QT's head computer:

```
ssh qtrobot@192.168.100.1
```

1. Install our project's dependencies:

```
git clone -b v1.0 https://github.com/robotpt/abm-setup ~/abm-setup
bash ~/abm-setup/scripts/pi_setup.bash
```

2. Increase the swap size, so we're able to build without running out of virtual memory:

   a. Turn off your swap memory:

```
sudo /sbin/dphys-swapfile swapoff
```

   b. Open your swap configuration file:

```
sudo nano /etc/dphys-swapfile
```

   c. Set *CONF_SWAPFACTOR* to 2 by changing the line that says `#CONF_SWAPFACTOR=2` to `CONF_SWAPFACTOR=2`, that is by deleting the `#` character to uncomment the line.

   d. Save and exit `nano` by hitting Ctrl+x, then typing 'y', and then hitting Enter twice to confirm things.

   e. Turn the swap file back on:

```
sudo /sbin/dphys-swapfile swapon
```

3. Clone our repositories and build them:

   a. Go to the source code directory in the catkin workspace:

```
cd ~/catkin_ws/src
```

   b. Clone our repositories:

```
git clone -b v1.0 https://github.com/robotpt/cordial
git clone -b v1.0 https://github.com/robotpt/qt-robot
```

   c. Build our workspace:

```
cd ~/catkin_ws
catkin_make
```

Note: It takes around five minutes for this command to finish. You can setup QT's body computer at the same time as it runs, if you like.

4. Setup our code to run when QT's head computer turns on.

   a. Copy the autostart script into the correct directory:

   ```
   roscp qt_robot_pi start_usc.sh /home/qtrobot/robot/autostart/
   ```

   b. Enable the autostart script:

      i. Open a webbrowser on QT (e.g., Firefox) and go to http://192.168.100.1:8080/.

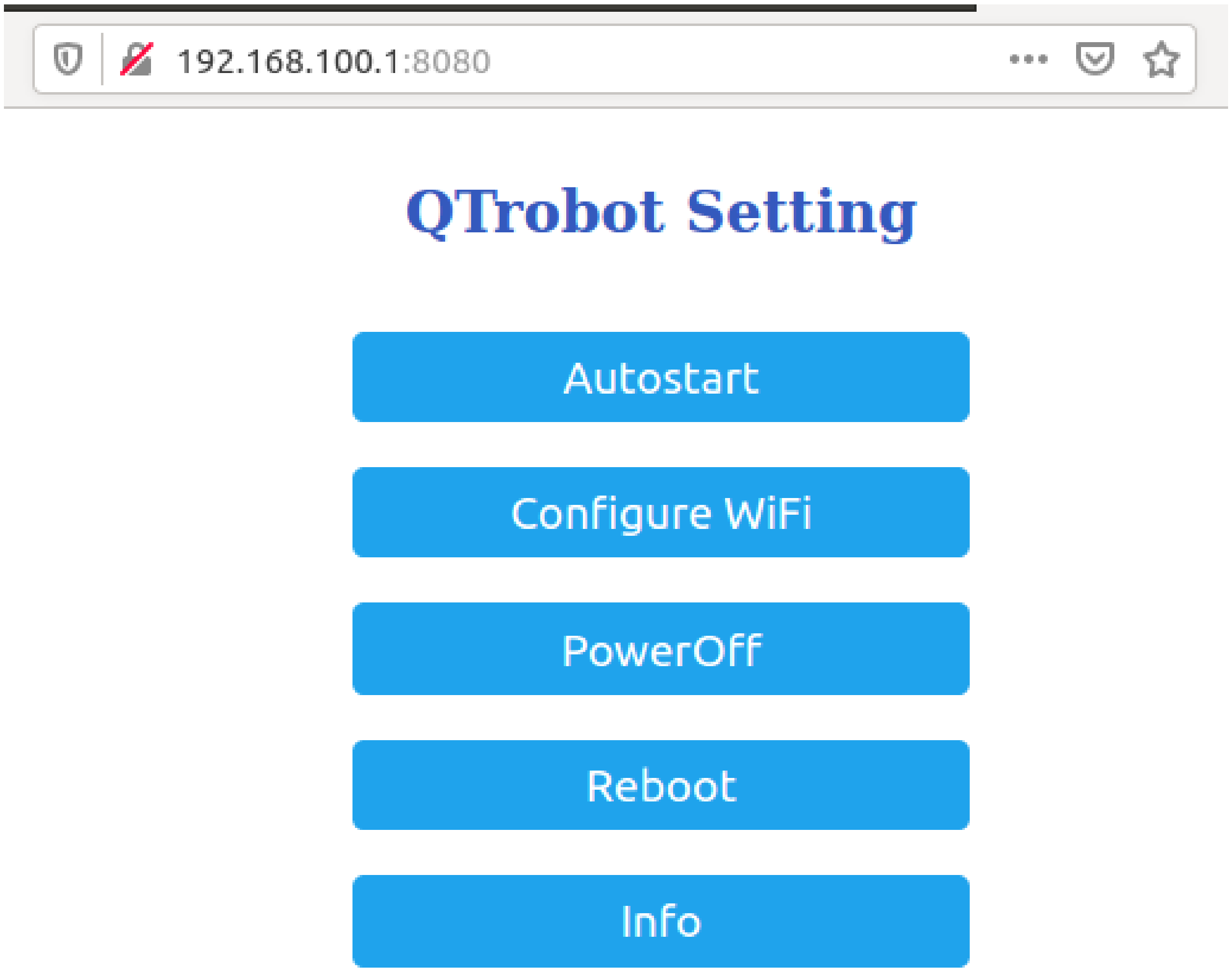


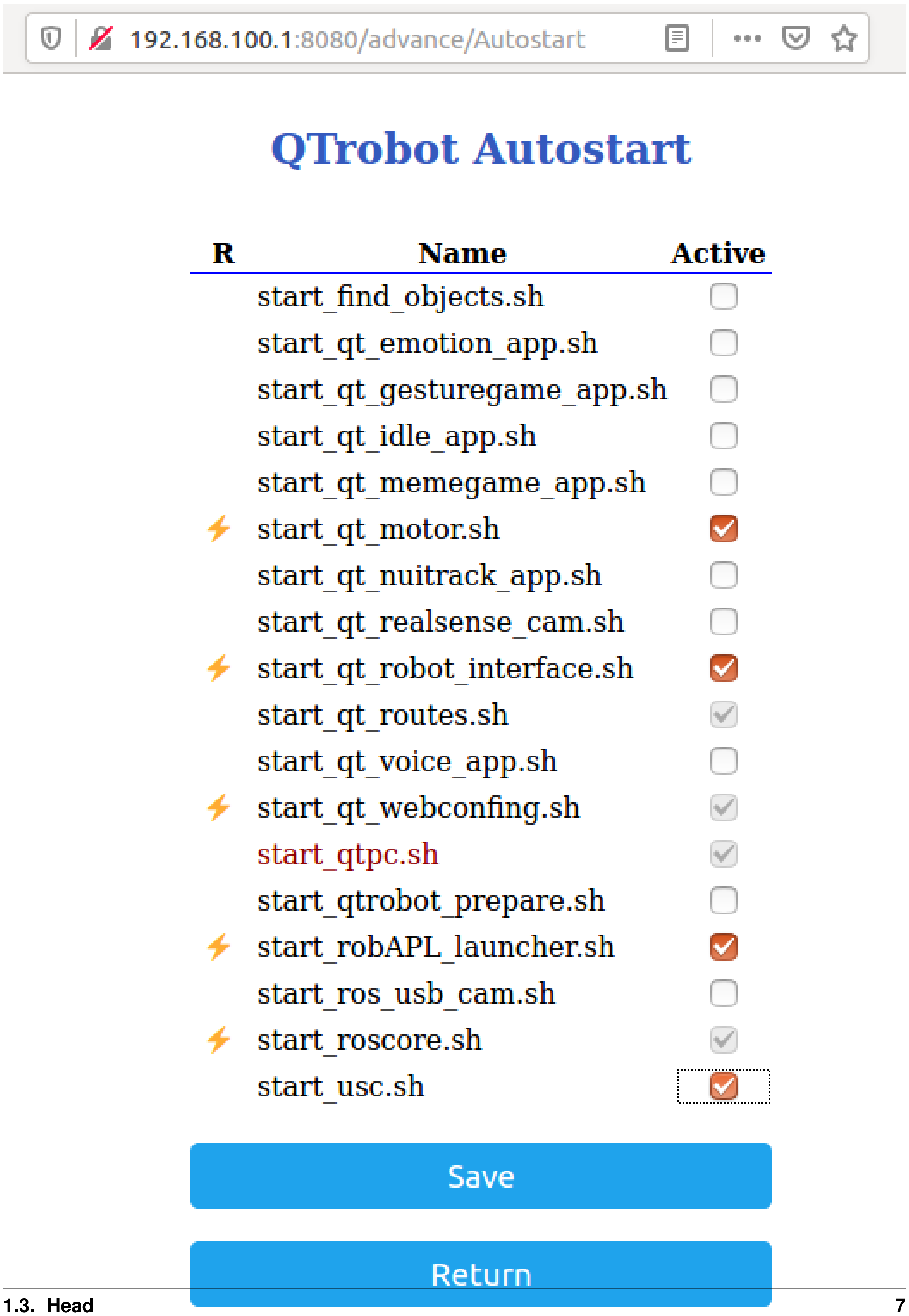
Fig. 1: QT's configuration menu.

      ii. Click 'Autostart'. You'll be prompted for a username and password. Enter `qtrobot` for both.

      iii. Click the 'Active' checkbox next to `start_usc.sh`.

      iv. Click 'Save' and then 'Return' twice.

Note: You can reboot to see these changes take effect, or continue on and we'll reboot eventually.



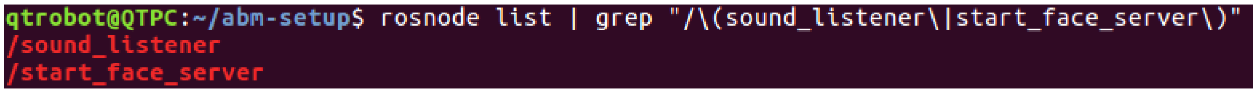
192.168.100.1:8080/advance/Autostart

QTrobot Autostart

R	Name	Active
	start_find_objects.sh	☐
	start_qt_emotion_app.sh	☐
	start_qt_gesturegame_app.sh	☐
	start_qt_idle_app.sh	☐
	start_qt_memegame_app.sh	☐
⚡	start_qt_motor.sh	✓
	start_qt_nuitrack_app.sh	☐
	start_qt_realsense_cam.sh	☐
⚡	start_qt_robot_interface.sh	✓
	start_qt_routes.sh	✓
	start_qt_voice_app.sh	☐
⚡	start_qt_webconfing.sh	✓
	start_qtpc.sh	✓
	start_qtrobot_prepare.sh	☐
⚡	start_robAPL_launcher.sh	✓
	start_ros_usb_cam.sh	☐
⚡	start_roscore.sh	✓
	start_usc.sh	✓

If you'd like, you can confirm that things are running after a reboot by opening a terminal and running the following command. You should see both `/sound_listener` and `/start_face_server`:

```
rosnode list | grep "/\(sound_listener\|start_face_server\)"
```



```
qtrobot@QTPC:~/abm-setup$ rosnode list | grep "/\(sound_listener\|start_face_server\)"
/sound_listener
/start_face_server
```

Fig. 3: What you should see if the head nodes are running correctly.

1.4 Body

1.4.1 Getting your Amazon Web Service credentials

For QT to speak, we use Amazon Polly, which requires an Amazon Web Services account. At our current usage, using Amazon Polly is free up to a certain level), but you will need a credit card to create an account.

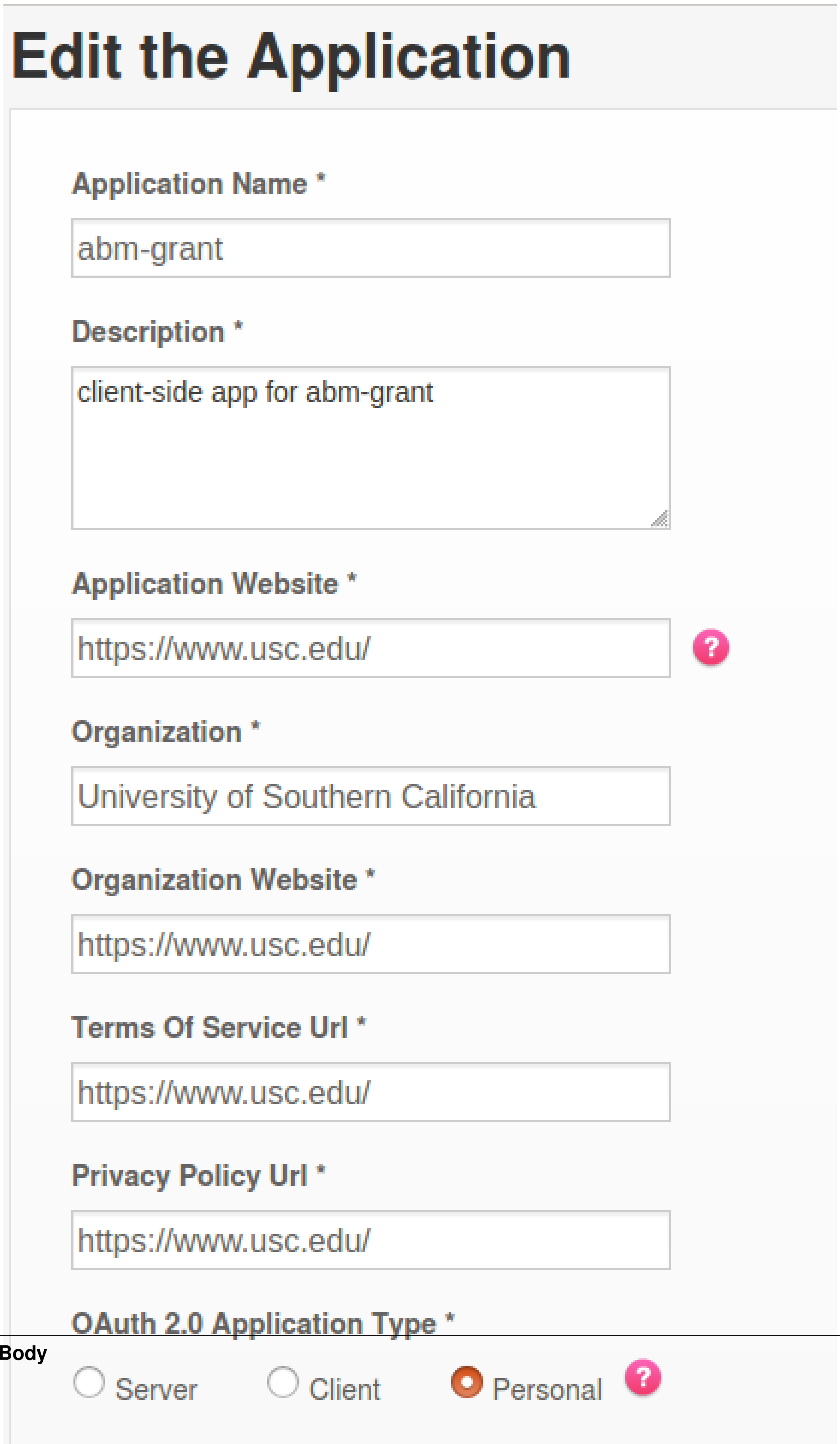
1. Create an Amazon Web Services account.
2. Once you sign in, in the top right of the page, click your account name (mine says "Audrow"), then in the drop-down menu click "My Security Credentials," then click "Create New Access Key."
3. Record your access key and keep it somewhere safe. You can do this by downloading this or just viewing it and copy-pasting it to somewhere for later reference.

Note: It is best practice to create separate accounts with less access than your root account and use those access keys, see Amazon's security best practices.

1.4.2 Getting your Fitbit credentials

You will need to make a Fitbit "app" for each Fitbit device. We are interested in the Client ID, Client Secret, and a generated code that saves us from having to login on a web browser.

1. Create a Fitbit account for each Fitbit device.
2. Login to your Fitbit account.
3. Go to register an app
4. Fill in the application. You can put whatever you think makes sense for most of them (URL, policy, etc.). (Make sure you include the *http* part int he urls.) The following are the parts that matter to get access to the Intraday data.
   - "OAuth 2.0 Application Type" should be "Personal"
   - "Callback URL" should be *http://localhost*
   - "Default Access Type" should be "Read-Only"
5. On the registered app's page, record your Client ID and Client Secret, and then click "OAuth 2.0 tutorial page," near the bottom.
6. On the Oauth2.0 tutorial page, set "Flow type" to "Authorization Code Flow."



Edit the Application

Application Name *

abm-grant

Description *

client-side app for abm-grant

Application Website *

https://www.usc.edu/

Organization *

University of Southern California

Organization Website *

https://www.usc.edu/

Terms Of Service Url *

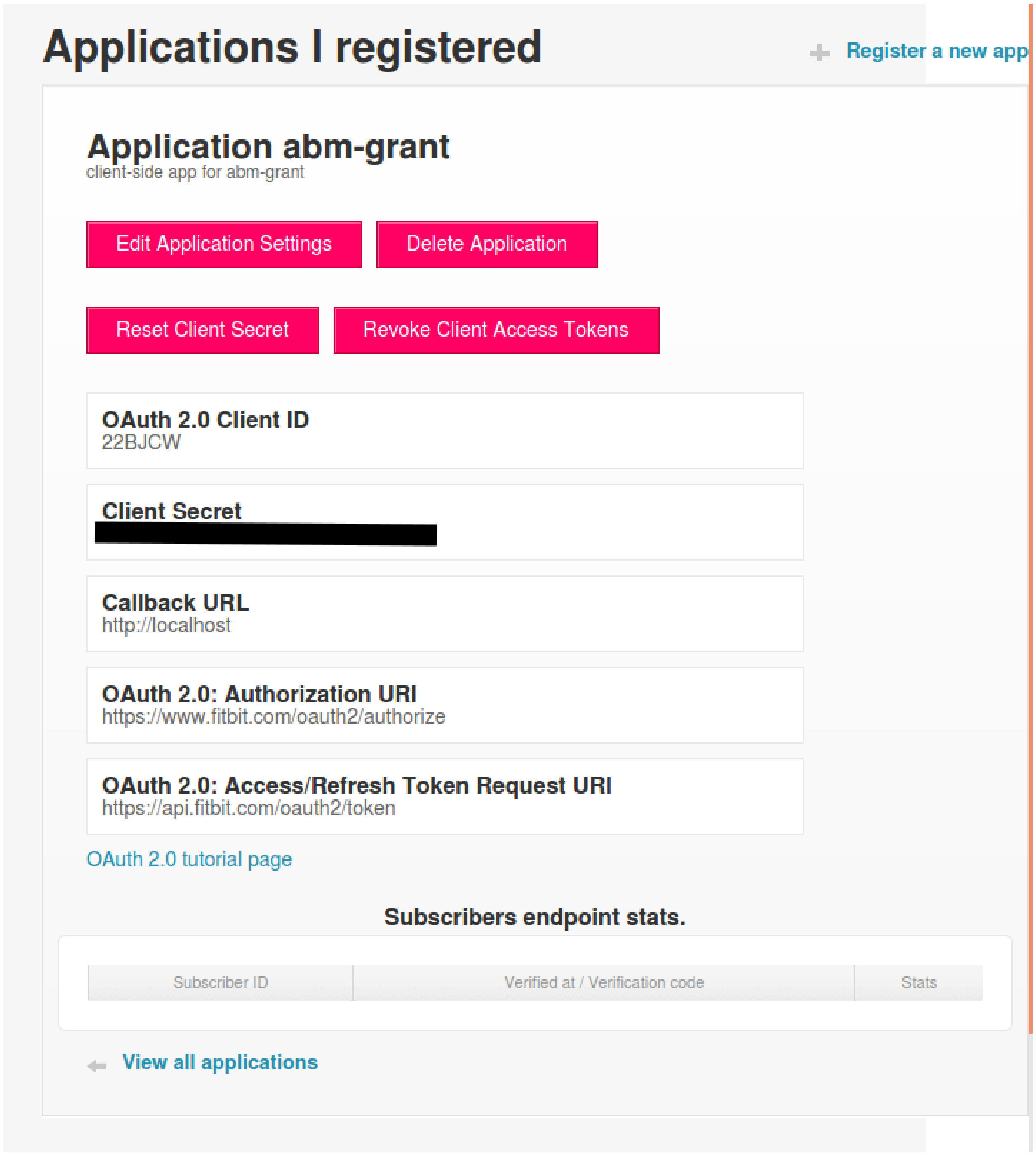
https://www.usc.edu/

Privacy Policy Url *

https://www.usc.edu/

OAuth 2.0 Application Type *

Server Client [✓ Personal]



Application abm-grant
client-side app for abm-grant

OAuth 2.0 Client ID
22BJCW

Client Secret
[redacted]

Callback URL
http://localhost

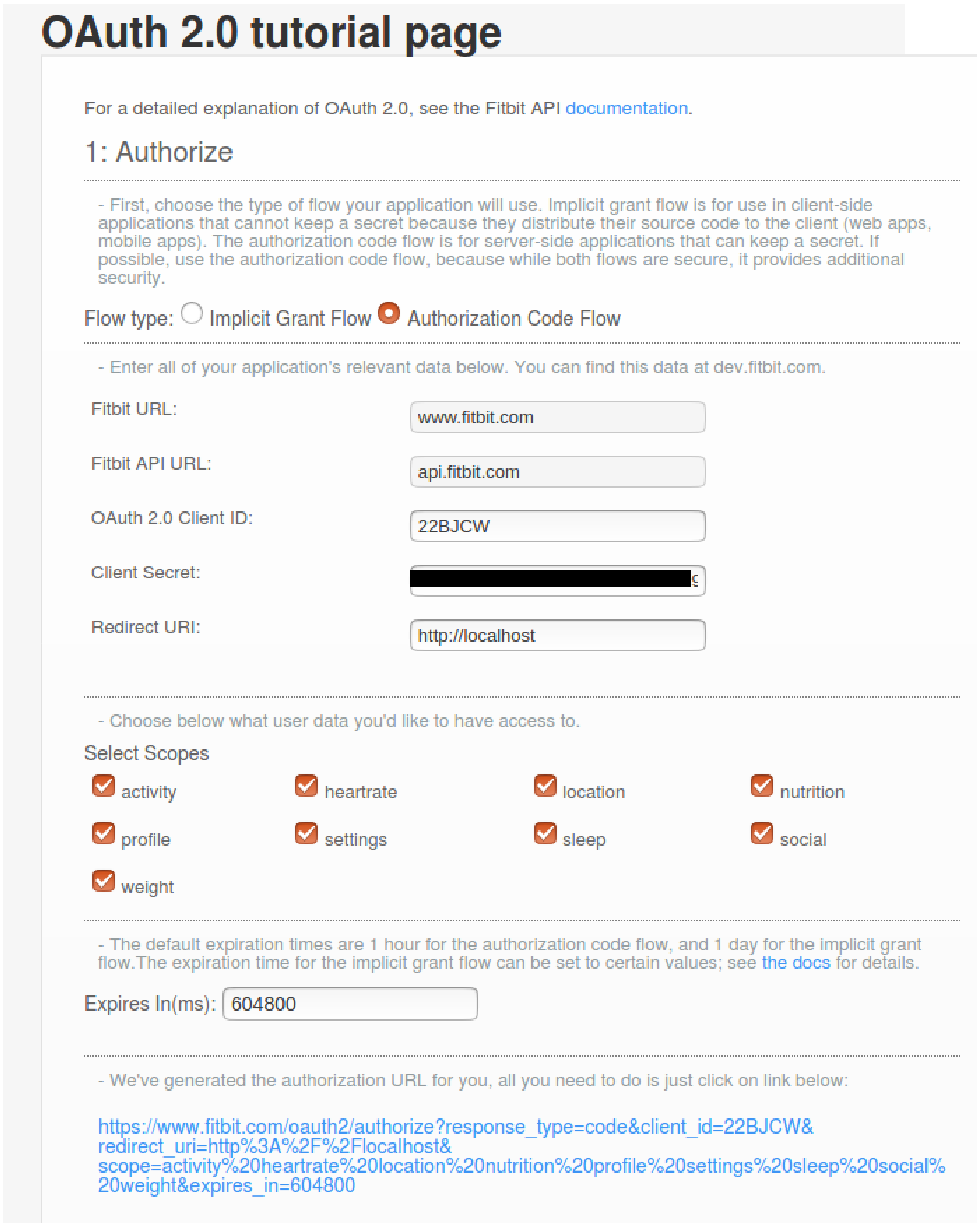
OAuth 2.0: Authorization URI
https://www.fitbit.com/oauth2/authorize

OAuth 2.0: Access/Refresh Token Request URI
https://api.fitbit.com/oauth2/token

Subscribers endpoint stats.

Subscriber ID	Verified at / Verification code	Stats

Fig. 5: The registered app page.



OAuth 2.0 tutorial page

For a detailed explanation of OAuth 2.0, see the Fitbit API documentation.

1: Authorize

- First, choose the type of flow your application will use. Implicit grant flow is for use in client-side applications that cannot keep a secret because they distribute their source code to the client (web apps, mobile apps). The authorization code flow is for server-side applications that can keep a secret. If possible, use the authorization code flow, because while both flows are secure, it provides additional security.

Flow type: Implicit Grant Flow [✓ Authorization Code Flow]

- Enter all of your application's relevant data below. You can find this data at dev.fitbit.com.

Fitbit URL:	www.fitbit.com
Fitbit API URL:	api.fitbit.com
OAuth 2.0 Client ID:	22BJCW
Client Secret:	[redacted]
Redirect URI:	http://localhost

- Choose below what user data you'd like to have access to.

Select Scopes

✓ activity	✓ heartrate	✓ location	✓ nutrition
✓ profile	✓ settings	✓ sleep	✓ social
✓ weight			

- The default expiration times are 1 hour for the authorization code flow, and 1 day for the implicit grant flow.The expiration time for the implicit grant flow can be set to certain values; see the docs for details.

Expires In(ms): 604800

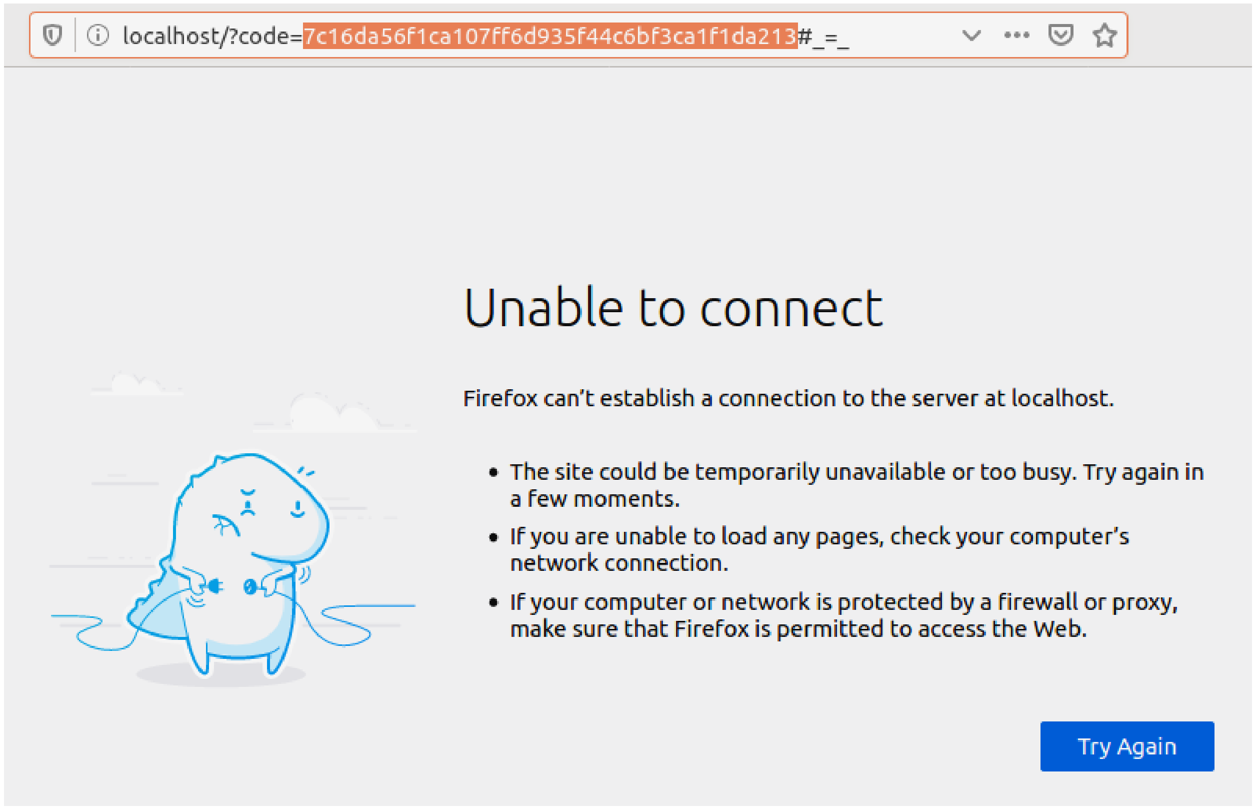
- We've generated the authorization URL for you, all you need to do is just click on link below:

https://www.fitbit.com/oauth2/authorize?response_type=code&client_id=22BJCW&redirect_uri=http%3A%2F%2Flocalhost&scope=activity%20heartrate%20location%20nutrition%20profile%20settings%20sleep%20social%20weight&expires_in=604800

Fig. 6: Oauth2.0 tutorial page with “Flow type” set to “Authorization Code Flow.”

Note: The "Expires In(ms)" text field is only used for "Implicit Grant Flow." "Authorization Code Flow," what we are using, expires in a fixed time (8 hours), but we are able to renew our authorization.

7. Click the URL above "1A Get Code." You'll be brought to an error page, but that's okay. We need the code from the URL. Record that code.



localhost/?code=7c16da56f1ca107ff6d935f44c6bf3ca1f1da213#_=_

Unable to connect

Firefox can't establish a connection to the server at localhost.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

Fig. 7: The page that you arrive at when clicking the URL above "1A Get Code." The code we are interested in in the URL is highlighted.

Warning: If the URL is longer than in the picture, go back to the OAuth2.0 tutorial page and make sure that you have the "Flow type" set to "Authorization Code Flow," not "Implicit Grant Flow."

Note: The code obtained in this step only works once. After you use it to initialize a Fitbit client, it cannot be used again. We use it to obtain an access and refresh token for talking to Fitbit's web API. If you need to reset Fitbit credentials for any reason, you will have to go to the OAuth2.0 tutorial page and get a new code.

Note: From this section, you should have the following information:

- Client ID
- Client Secret

- A generated code

1.4.3 Setting up our interaction

0. Change your system timezone to be in your current timezone. To do this, you can click the time in the upper-right of the desktop on QT and then click 'Time & Date settings. . . '
1. Open a terminal and clone this repository onto QT's body computer:

   ```
   git clone -b v1.0 https://github.com/robotpt/abm-setup ~/abm-setup
   ```

2. Run a script to allow for updates:

   ```
   sudo bash ~/abm-setup/scripts/nuc_setup.bash
   ```

Note: This step takes five minutes or so.

3. Setup Docker:

   a. Install Docker:

   ```
   curl -fsSL https://get.docker.com -o get-docker.sh
   sh get-docker.sh
   ```

   b. Set Docker to run without `sudo`:

   ```
   sudo groupadd docker
   sudo gpasswd -a $USER docker
   newgrp docker
   ```

   c. Test that Docker is installed correctly and works without `sudo`:

   ```
   docker run hello-world
   ```

4. Setup Docker-compose:

   a. Install Docker-compose:

   ```
   sudo curl -L "https://github.com/docker/compose/releases/download/1.25.3/
   ↪docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
   sudo chmod +x /usr/local/bin/docker-compose
   ```

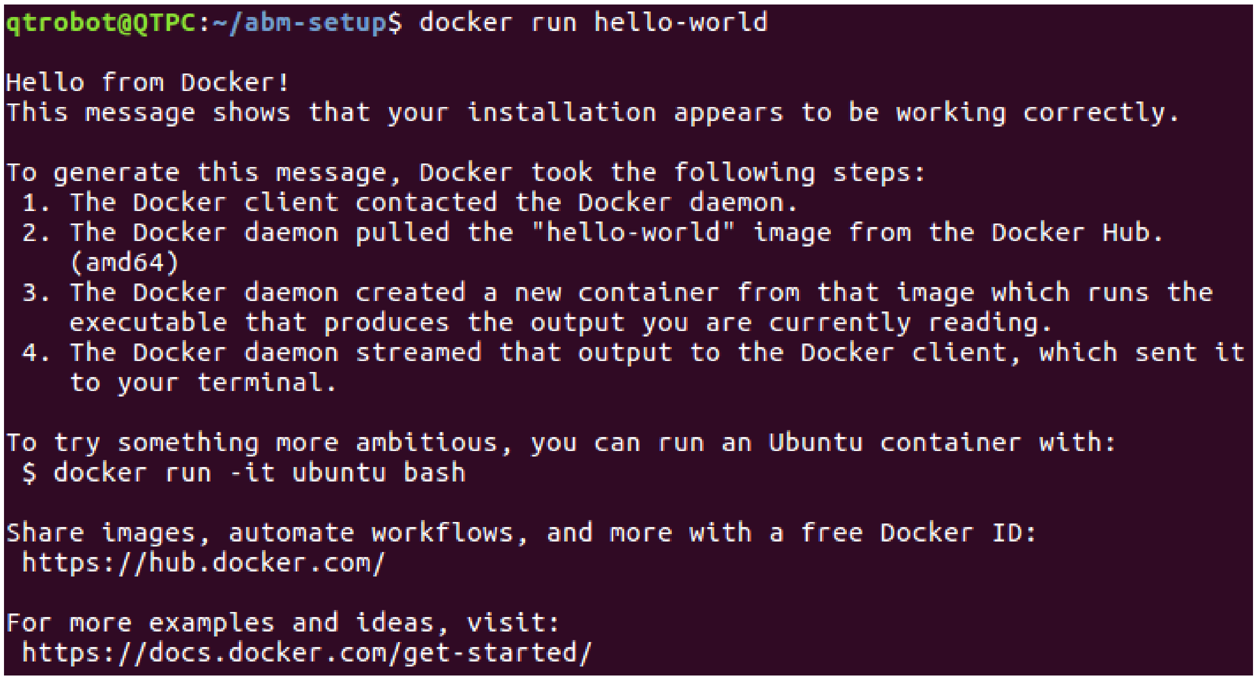
   b. Check that docker compose is installed correctly:

   ```
   docker-compose version
   ```

5. Setup the docker container:

   Note: The first time that you run the Docker script, it will take around 15 minutes to setup the container. After that, it will be fast. Feel free to take a break or go get coffee :-)

   a. Go to the `docker` directory in the `abm-setup` folder:



```
qtrobot@QTPC:~/abm-setup$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Fig. 8: What is printed from running the `hello-world` docker container.

```
cd ~/abm-setup/docker
```

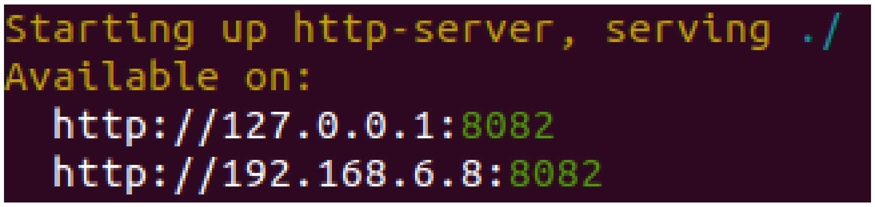
b. Run the `docker.sh` script with the `setup` option:

```
bash docker.sh setup
```

Note: I did have an error occur during this command one of the times I was setting it up. It might have been a network issue. I ran it again and it succeeded. If you have trouble here let me know.

c. Enter your Fitbit and Amazon Web Services credentials as prompted.

d. You will then be shown the URLs where the tablet GUI will be hosted. There will be a few of them. We want one that starts with "192", rather than "127" or "10", because it will accept connections from other devices on the local network. Write down the relevant address.



```
Starting up http-server, serving ./
Available on:
  http://127.0.0.1:8082
  http://192.168.6.8:8082
```

Fig. 9: An example of the URLs that that the interaction will use. In this case, we want to write down `http://192.168.6.8:8082`.

Note: If you don't see an address with "192" at the beginning, try changing QT to a different wireless network.

e. Hit Ctrl+C to close the container.

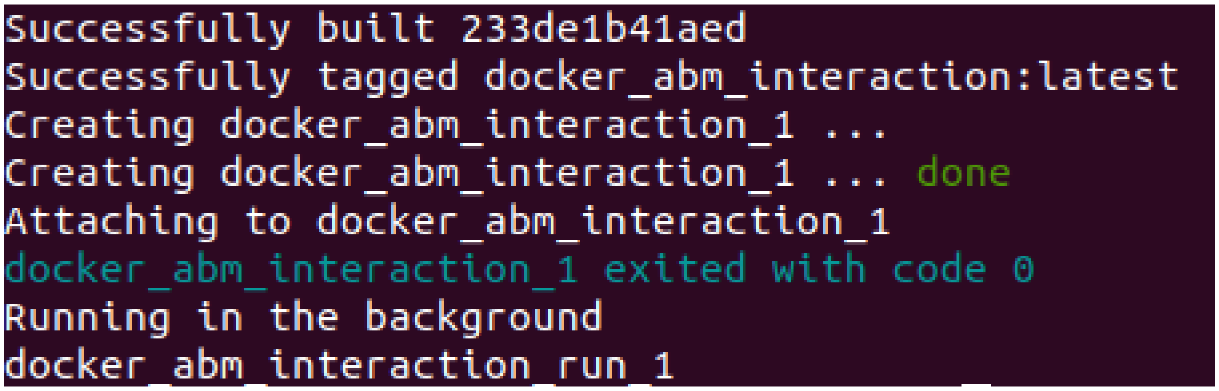
6. Run the interaction:

a. Make sure that you're in the `docker` directory in the `abm-setup` folder:

```
cd ~/abm-setup/docker
```

b. Run the `docker.sh` script with the `run` option:

```
bash docker.sh run
```



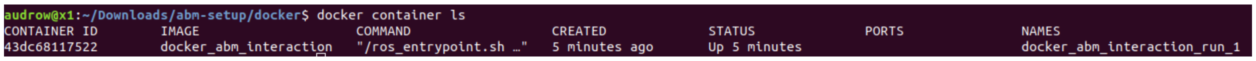
```
Successfully built 233de1b41aed
Successfully tagged docker_abm_interaction:latest
Creating docker_abm_interaction_1 ...
Creating docker_abm_interaction_1 ... done
Attaching to docker_abm_interaction_1
docker_abm_interaction_1 exited with code 0
Running in the background
docker_abm_interaction_run_1
```

Fig. 10: An example of the final message after the interaction run script.

c. Make the interaction run on startup:

i. List your Docker containers:

```
docker container ls
```



```
audrow@x1:~/Downloads/abm-setup/docker$ docker container ls
CONTAINER ID        IMAGE                    COMMAND                  CREATED             STATUS              PORTS               NAMES
43dc68117522        docker_abm_interaction   "/ros_entrypoint.sh …"   5 minutes ago       Up 5 minutes                            docker_abm_interaction_run_1
```

Fig. 11: An example of running containers.

ii. Copy the "CONTAINER ID".

iii. Update the container's restart policy:

```
docker container update --restart=unless-stopped <YOUR COPIED␣
↪CONTAINER ID>
```

Note: At this point, you should reboot QT. You can do this by either pushing the button on the back of QT or typing `sudo reboot` into the head computer's terminal.

To test that things are setup correctly, you can take the URL for the GUI that you wrote down and type it into the web-browser on any device that's on the same network. QT should begin asking you about your name, if it is your first interaction.

1.5 Tablet

For either tablet supplied by LuxAI with QT, or any Android tablet for that matter, we’re going to set up the tablet to run as a Kiosk using the app Fully Kiosk Browser.

1. Sign on to the Google Play Store.
2. Search for and download *Fully Kiosk Browser*.
3. Start *Fully Kiosk browser* and set the start URL to the GUI URL that you wrote down earlier.
4. Adjust settings in *Fully Kiosk browser*:
   i. In ‘Settings > Web Zoom and Scaling’, disable ‘Enable Zoom’
   ii. In ‘Settings > Web Auto Reload’, set ‘Auto Reload after Page Error’ to ‘2’.

With this app, you can make it so that it’s challenging to get out of the app or do other things on the tablet. You can go into ‘Settings > Kiosk Mode (PLUS)’ to play with these settings. A plus license is 6.90 EUR per device (about 7.50 USD).

CHAPTER 2

Known issues

2.1 Critical

2.2 Not critical

- With rare chance, audio may be skipped (PyAudio error)
- Sometimes beginning of sound is cut off

2.3 Little fixes

CHAPTER 3

Planned changes in the interaction

- Sync Fitbit during a conversation if they choose to
- Option to checkin early if they meet their walk goal
- Make the GUI timeout occur a set period of time after QT finishs speaking

CHAPTER 4

Suggestions / Bug report

For suggested improvements or bugs to report, please email Audrow Nash at audrow.nash@gmail.com.