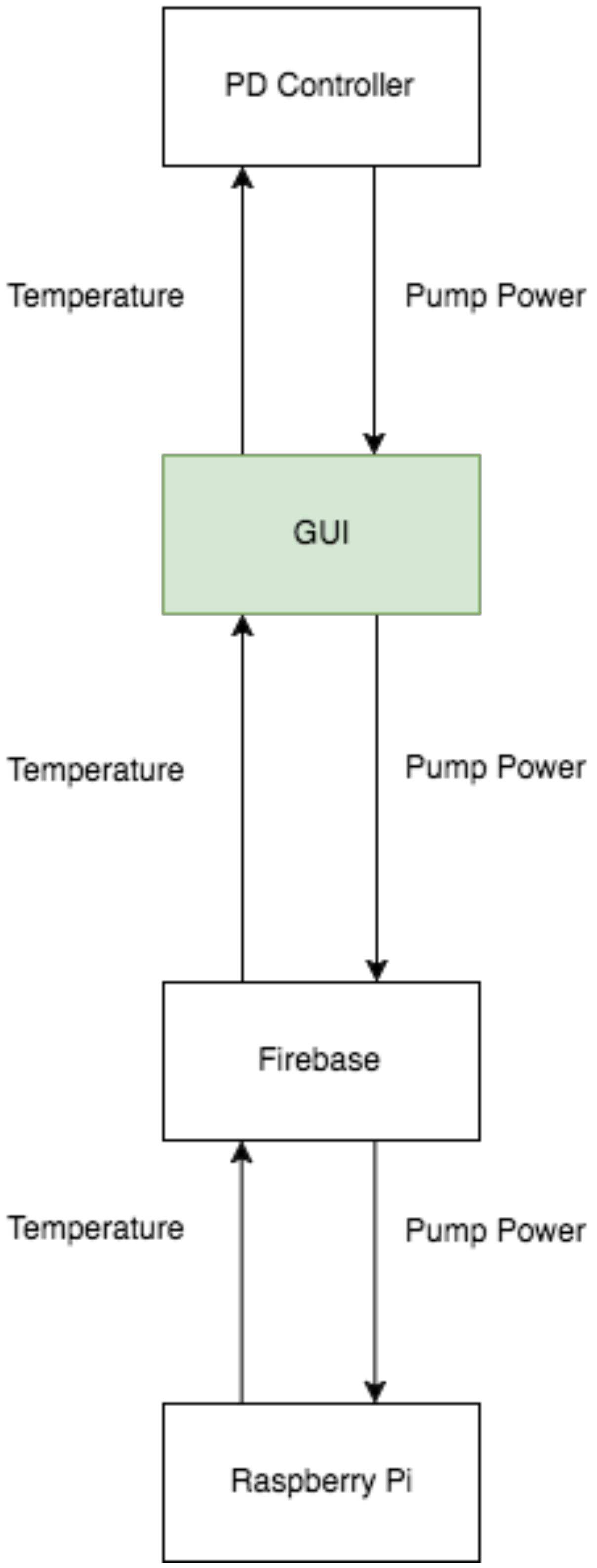
ARCHITECTURE OVERVIEW

FILES STRUCTURE		
	gui/	
	main.py	starts everything (always starts gui)
	simultion.py	starts simulation
	logic/	contains Controller
	sim_scripts/	Other scripts for simulation

NORMAL MODE



PD Controller

Temperature

Pump Power

GUI

Temperature

Pump Power

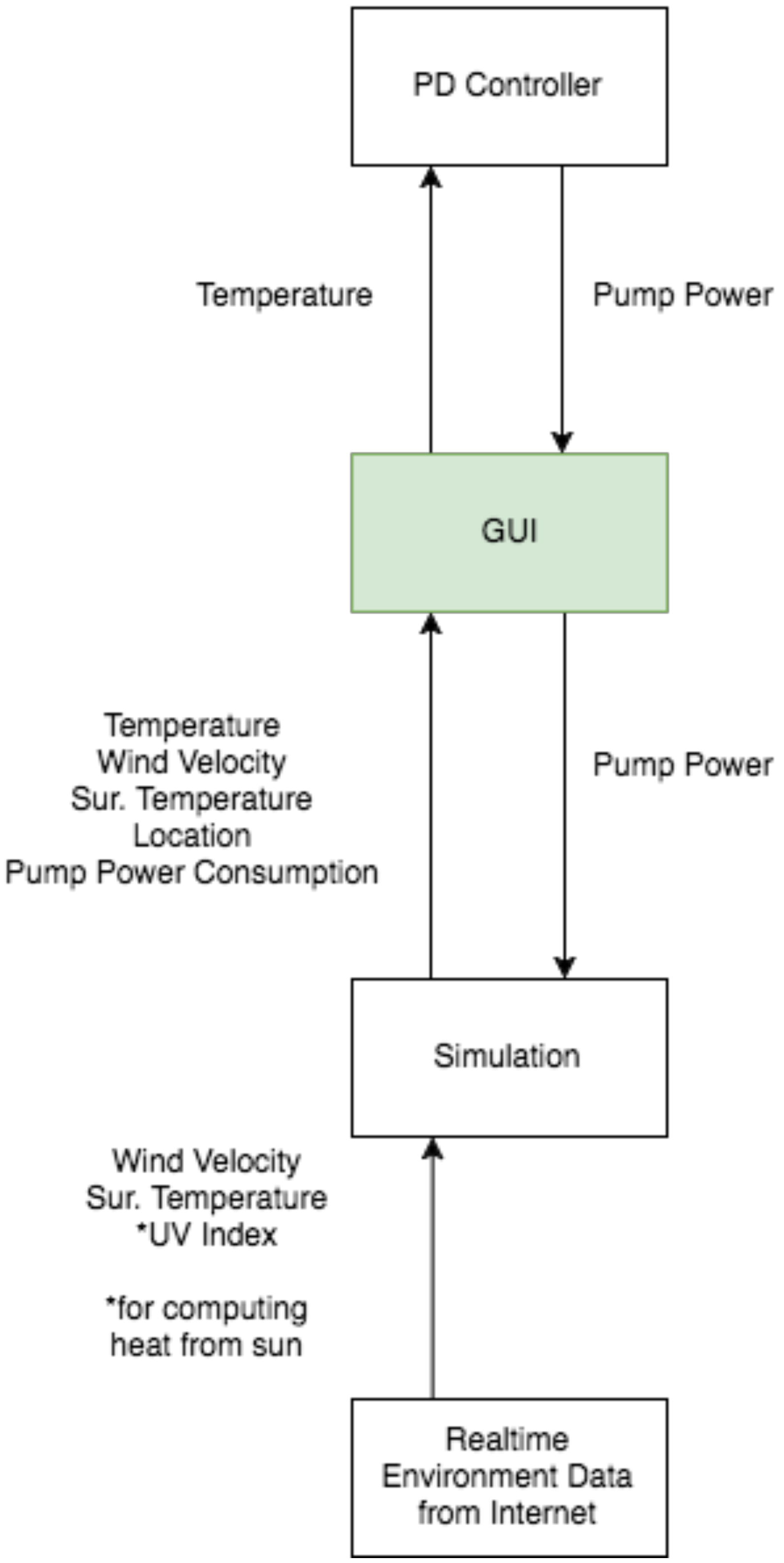
Firebase

Temperature

Pump Power

Raspberry Pi

SIMULATION MODE



PD Controller

Temperature

Pump Power

GUI

Temperature
Wind Velocity
Sur. Temperature
Location
Pump Power Consumption

Pump Power

Simulation

Wind Velocity
Sur. Temperature
*UV Index

*for computing
heat from sun

Realtime
Environment Data
from Internet

EXTRA INFO

Threading

We don’t want the rest of our code to be blocked by code that updates firebase / gets real time data. We want to do these things at the same time.

Threading is a way of doing this.

State (not state machine)

State is the data your program is operating on (e.g. current temperature). We use 3 stores of state in this app.

1. Firebase. Used as a means of communication between the laptop and the raspberry pi.

2. The GUI `State` widget (`gui/State.py`). Holds the state for the entire app. This is not how people normally use Kivy, but it allows for better modularity (we can separate widgets better without overly nesting them)

3. The Simulation State. This is used as a means of communication between GUI and Simulation.

Simulation vs Normal Mode

In `gui/constants.py`,
SIM_MODE = True -> GUI runs in Simulation Mode
SIM_MODE = False -> GUI runs in Normal Mode

Realtime Graph

To display, used kivy garden graph package (basically something someone else wrote for kivy).

For the graph data, we created a History class (in `gui/State.py`) to keep track of a configurable number of temperature and power data points over time.

Real Environment Data

The simulation grabs realtime data from data.gov.sg and uses this to compute the heat transfers. Auto-updates every minute.