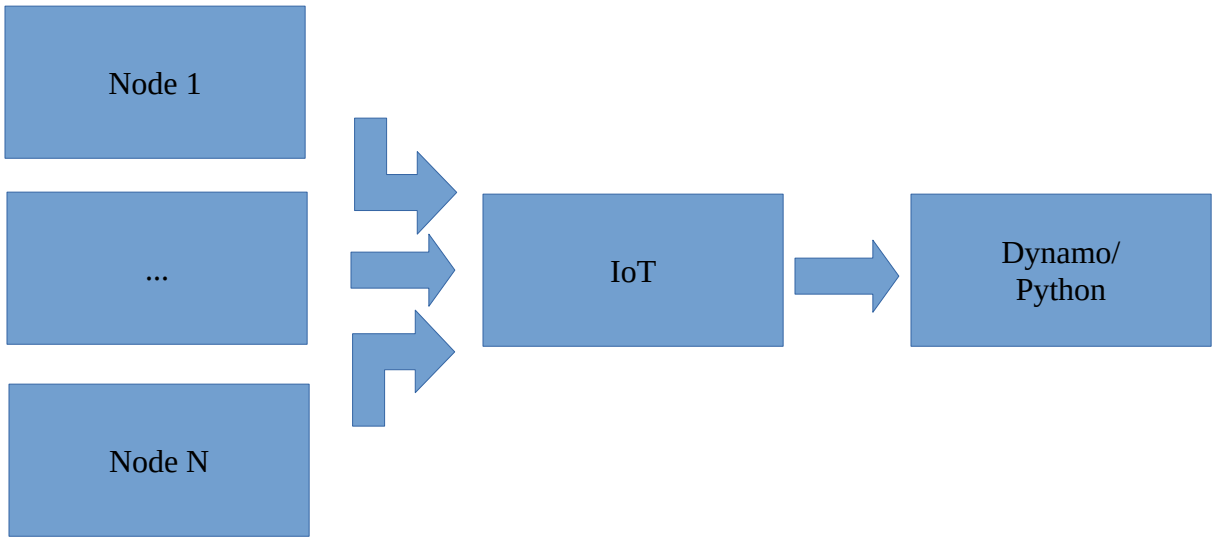
Digital Twin Project

The system goal is monitor temperature and humidity inside a building and send the data to an IoT cloud. The data can them be retrieved and analyzed further.

Information flow



IoT cloud

ThingSpeack by Mathworks
ThingSpeack stores data in channels and fields. Every account gets a certain amount of channels (depending on the license), each channel has 8 fields.
The logic is as follows: a channel corresponds to a particular device and a field is dedicated to a particular flow of information.
For example:
If there are 2 nodes, each has 3 sensors attached to it. ThingSpeack engineers propose the following: Node 1 sends data to Channel 1, sensor 1 readings go to field 1 of channel 1, sensor 2 readings go to field 2 of channel 1 and sensor 3 readings go to field 3 of channel 1.
At the same time, Node 2 sends data to Channel 2, sensor 1 readings go to field 1 of channel 2, sensor 2 readings go to field 2 of channel 2 and sensor 3 readings go to field 3 of channel 2. And so on, and so on.

In reality a channel can receive data from several sources, however, some functionality will be lost. For this project the structure described in the example above fits better and thus will be adopted.
As of now we have a free account that comes with the following limitations:
1) 4 Channels
2) Max update rate 1/15 seconds
3) 3 million messages / year
4) Channels have to be Public

	FREE For small non-commercial projects	ACADEMIC For academic use by faculty, staff, or researchers at degree-granting institutions[1]
Scalable for larger projects	✖ No. Annual usage is capped.	✔
Number of messages	3 million/year (~8,200/day)[2]	33 million/year per unit (~90,000/day per unit)[2]
Message update interval limit	Every 15 seconds	Every second
Number of channels	4	250 per unit
MATLAB Compute Timeout	20 seconds	60 seconds
Private channel sharing	Limited to 3 shares	Unlimited
Technical Support	Community Support	Standard MathWorks support

Academic license price: USD 275 *unit*/year

Node
A node suppose to measure Temperature and Humidity in its direct vicinity and send the data to the cloud.
MCU: ESP8266
3.3 V board

Sensor: DHT11 (This is the one I have at hand)

- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 20-80% humidity readings with 5% accuracy
- Good for 0-50°C temperature readings ±2°C accuracy
- No more than 1 Hz sampling rate (once every second)

The ranges of the sensor are problematic. Humidity might easily go above 80%, as a matter of fact it was higher than 80% for the most of the last night.

It would be better to get DHT22

- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 0-100% humidity readings with 2-5% accuracy
- Good for -40 to 80°C temperature readings ±0.5°C accuracy
- No more than 0.5 Hz sampling rate (once every 2 seconds)

The ranges are higher. The sampling rate is lower, but still well within our requirements.

Power source:
Still at the design stage. Hopefully 4 AA batteries per node.