PetFeeder

a smart pet feeding system

by Ben Solomon & Aviv Yonai

Summary

PetFeeder is a smart feeding system containing Hardware product, Azure web services and a Mobile App. The system is designed to enable users to perform smart pet feeding from anywhere at any time.

What is it good for?

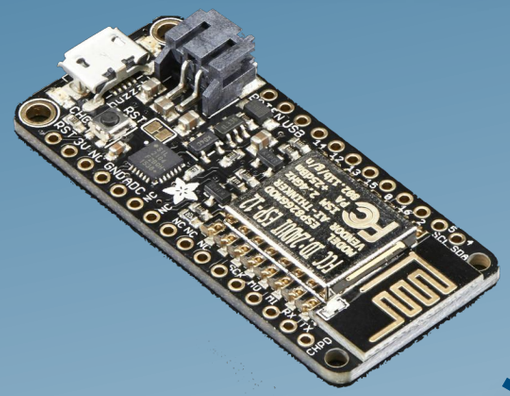
During our stressful everyday routine we sometimes forget to
take care of our 4-legged friends who are waiting at home.
Did you ever had to cancel or postpone your plans just because you have to go home and feed your pet? not anymore. with PetFeeder users are able to create a regular-base feeding schedule or feed their furry friend in real time just by pressing a button.

Main Features

1. Online system status
2. Manual feed
3. Automatic feed by timer
4. Feeding verification
5. Eating detection

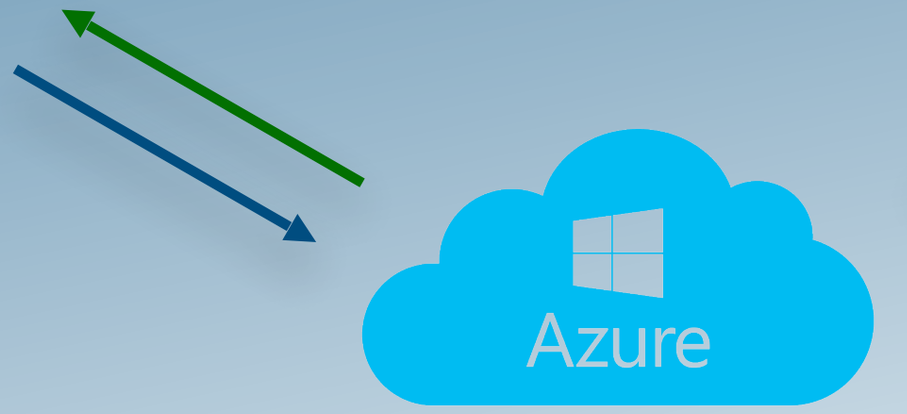
How does it work?

Through Azure services the hardware is able to communicate with the PetFeeder App and Vice versa.



Adafruit Feather
HUZZAH ESP8266

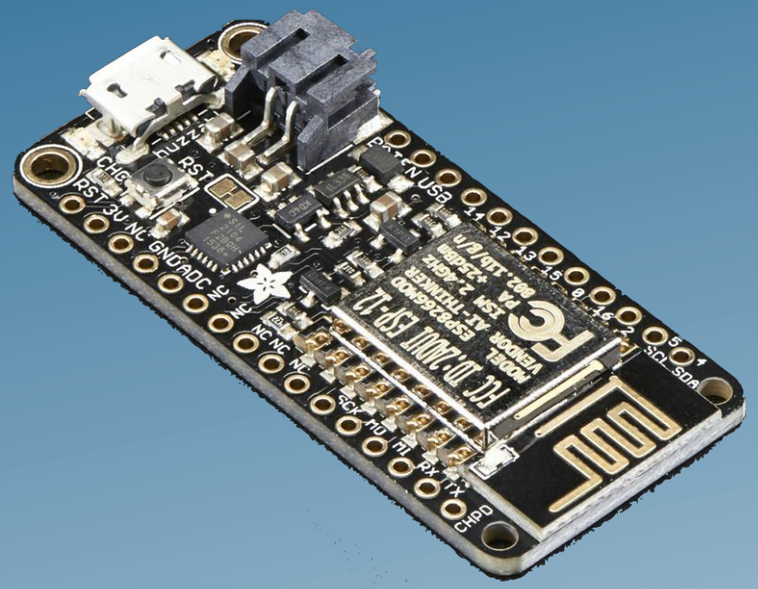
Mobile App



Azure Web Services

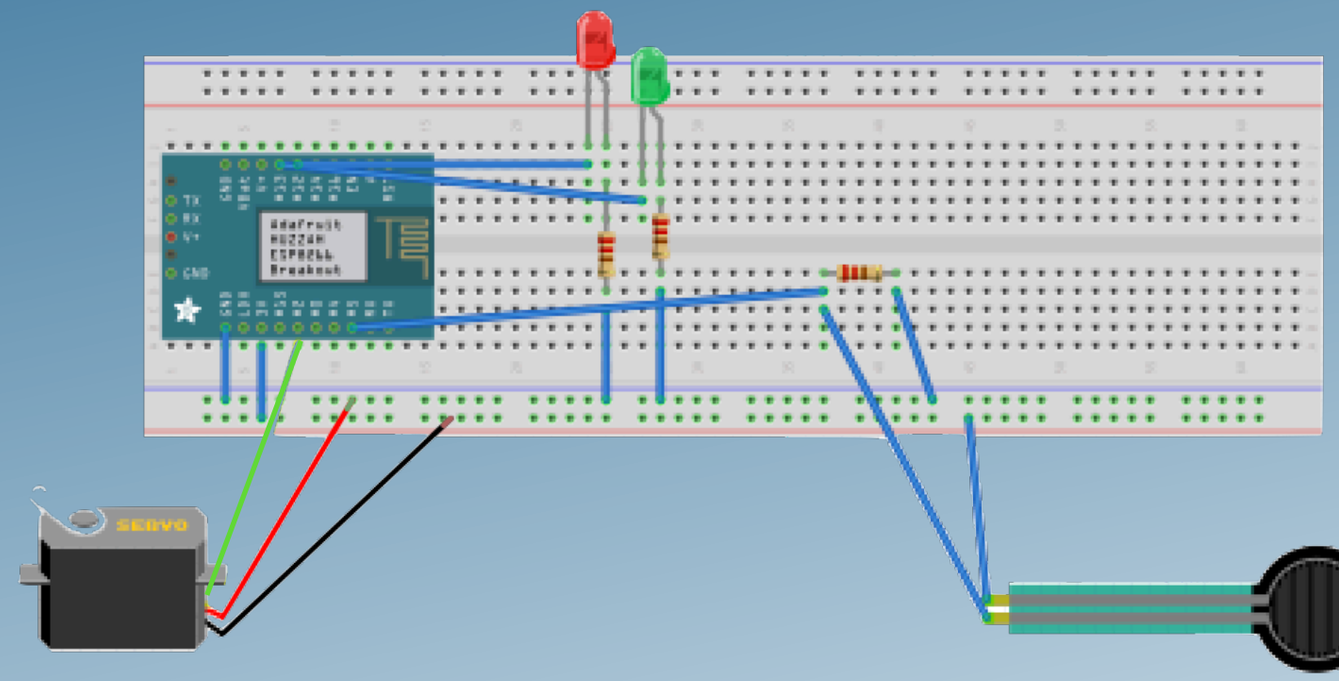
Hardware

Our Board:



The Adafruit Feather HUZZAH ESP8266 is an 'all-in-one' ESP8266 WiFi development board with built in USB and battery charging.

Control LEDs



Servo Motor

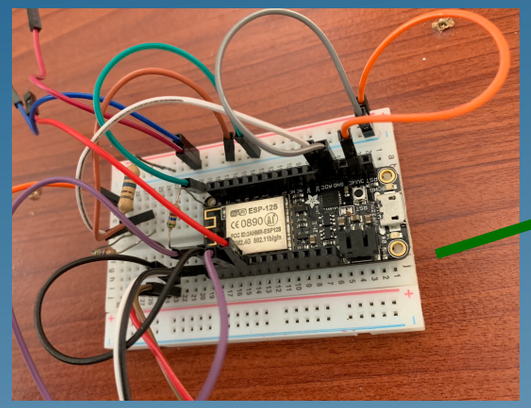
- Spins the cap that pours the food when its feeding time

FlexiForce Force Sensor

- Connected to the food platter and sends the user the current weight.
- Enables verifying and monitoring the feeding process

PetFeeder

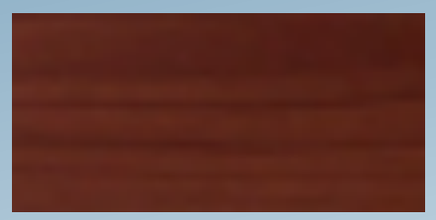
What its made of?



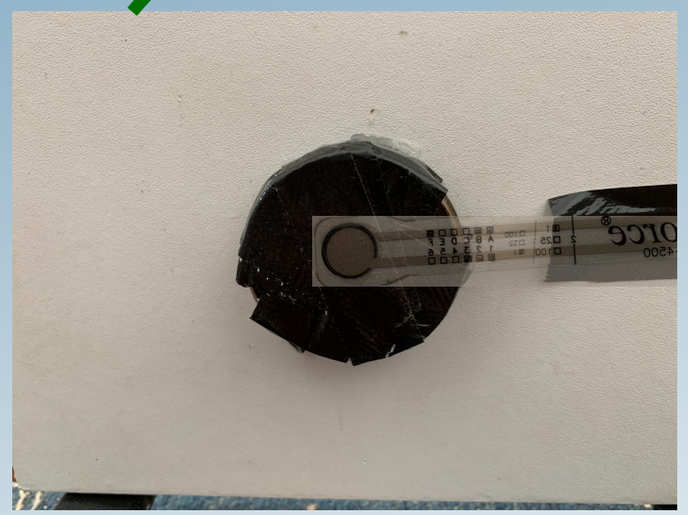
ESP8266
Board

Feeding bottle

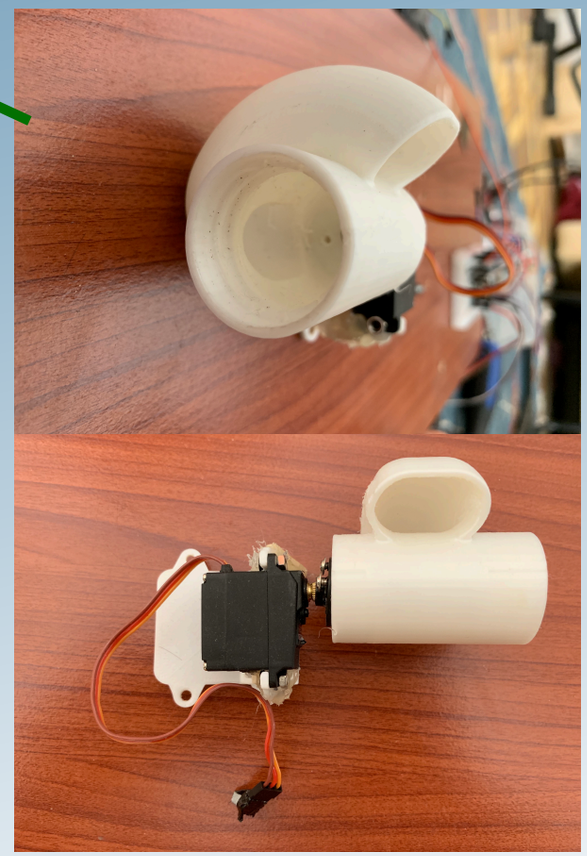
Plastic box



Old drawer



Pressure Sensor



3D Printed cap attached
to Servo Motor

Azure Web Services

Azure Functions

- **GetPlateWeight**
- **UpdatePlateWeight**
- **MonitorWeight**
- **Negotiate:** Establishes Connection with SignalR
- **FeedbyTime:** Time-Triggered feeding
- **MessageReciever:** Gets messages from devices and calls updating-DB function

Azure Tables

- Users Table
- Devices Table
- Feeding Times Table
- History Log Table
- Weights Table

IOT Hub

- Connected to the Devices and enables communication with them

Azure

Azure Web Services

DB - Azure Tables

Users Table

Device Code (Partition Key)	User's ID (Row Key)

Devices Table

Device Code (Row Key)	Pet Name	Pet Type

Feeding Times Table

Device Code (Partition Key)	Feed Timer

History Log Table

Device Code (Partition Key)	User's ID (Row Key)	ActionType	Timestamp

Weights Table

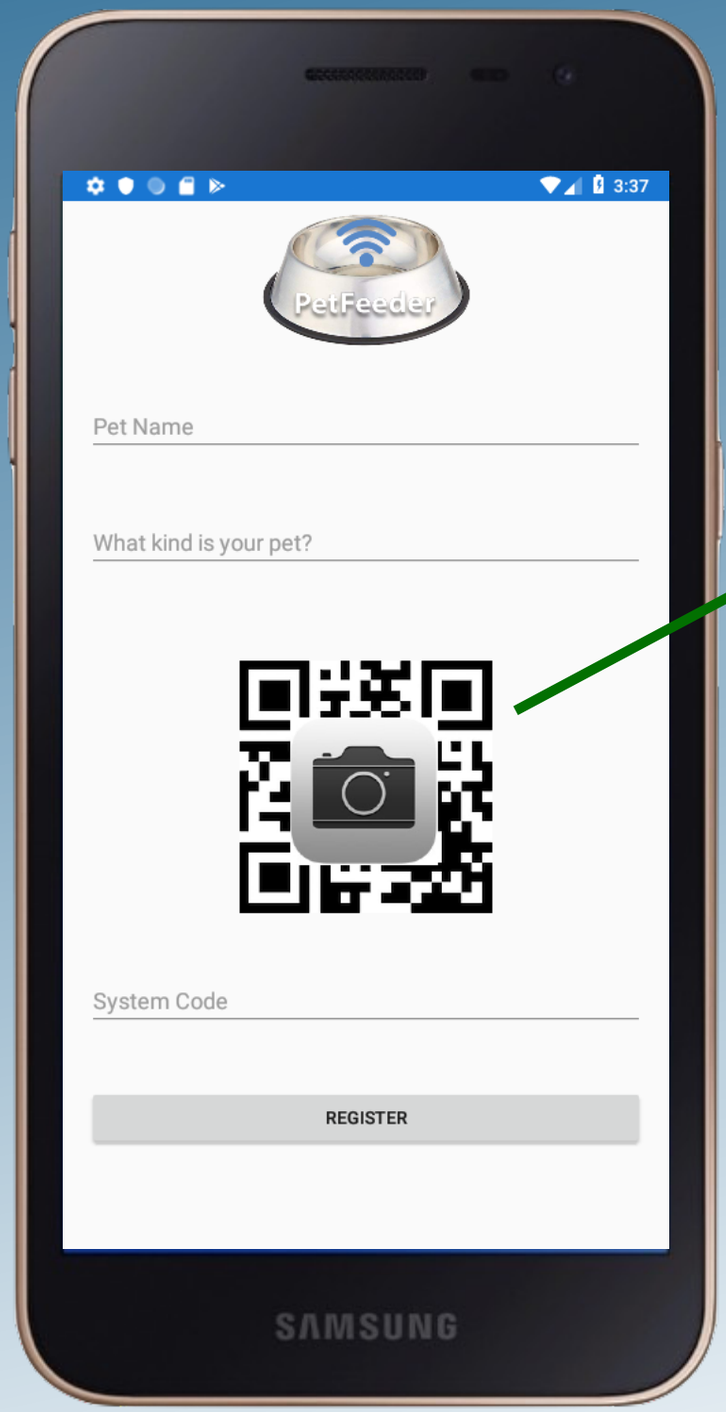
Device Code (Partition Key)	MeasurementType	Weight

Mobile App

- Developed with Xamarin
- Using Model-View method
- Android app and iPhone app are both available

First Login Page

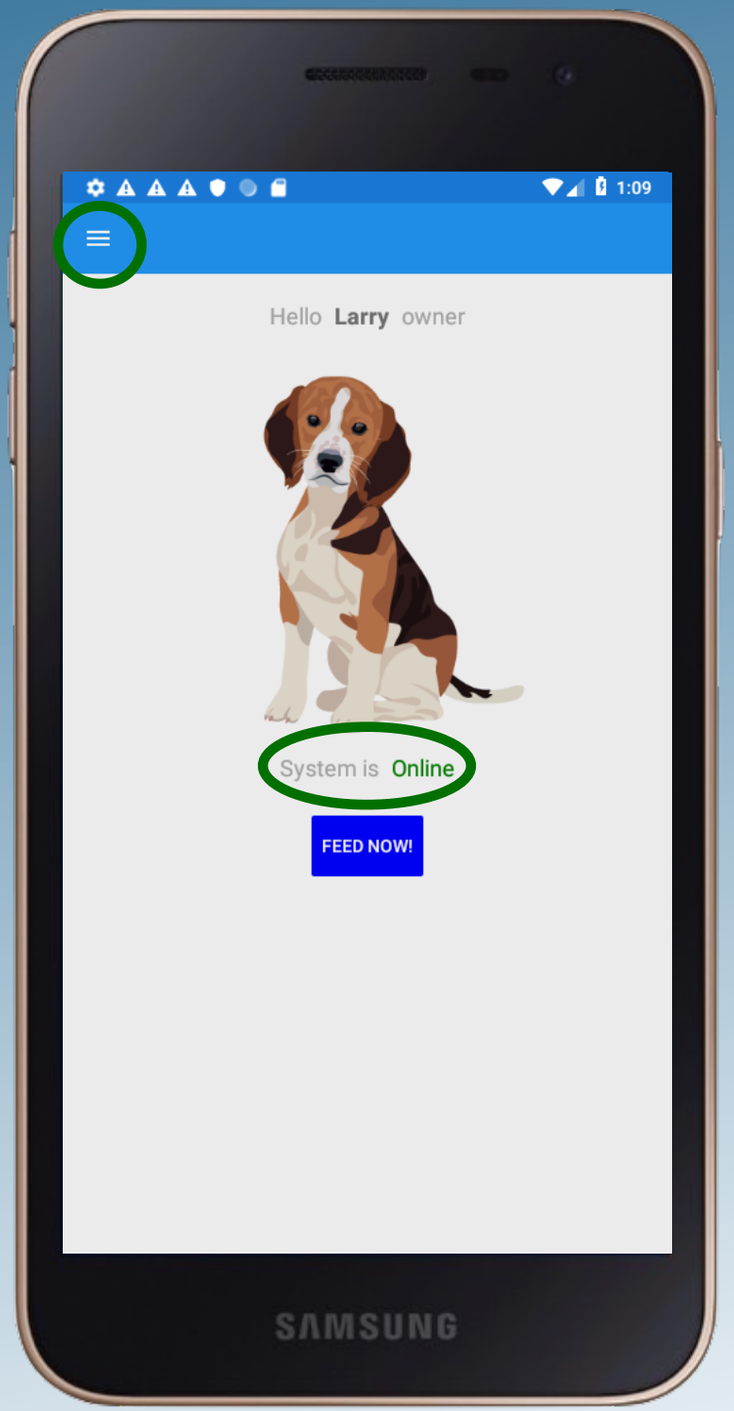
Appears on the first use of the App, or when Registering a new Device



Every Device has its own unique System Code implemented in it’s QR-Label

Main Page

- Contains the “FEED!” button which creates a real-time feeding instantly
- System status - indicates whether the_device is online and functioning, or disabled.
- App Side-Menu button



Main Page
System Status update

Message Reader
IoTHubTrigger Function

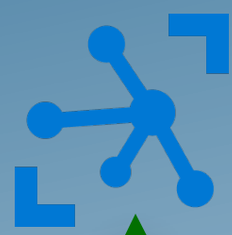
Sending SignalR message

Listen to SignalR messages to update connection status

Status is Offline if no messages within 10 seconds

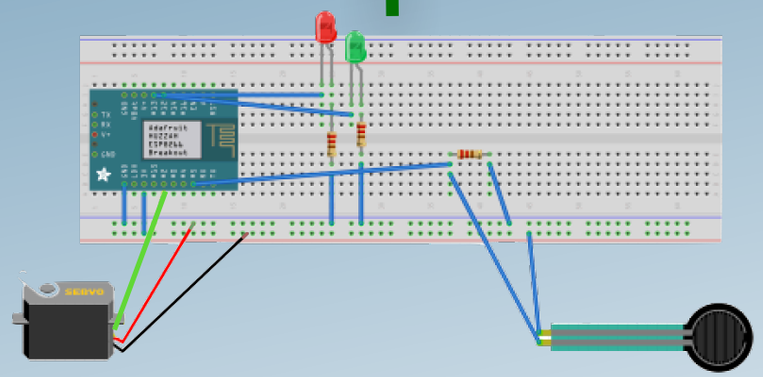
Connects negotiate http function to get HubConnection object

Negotiate
SignalR Function



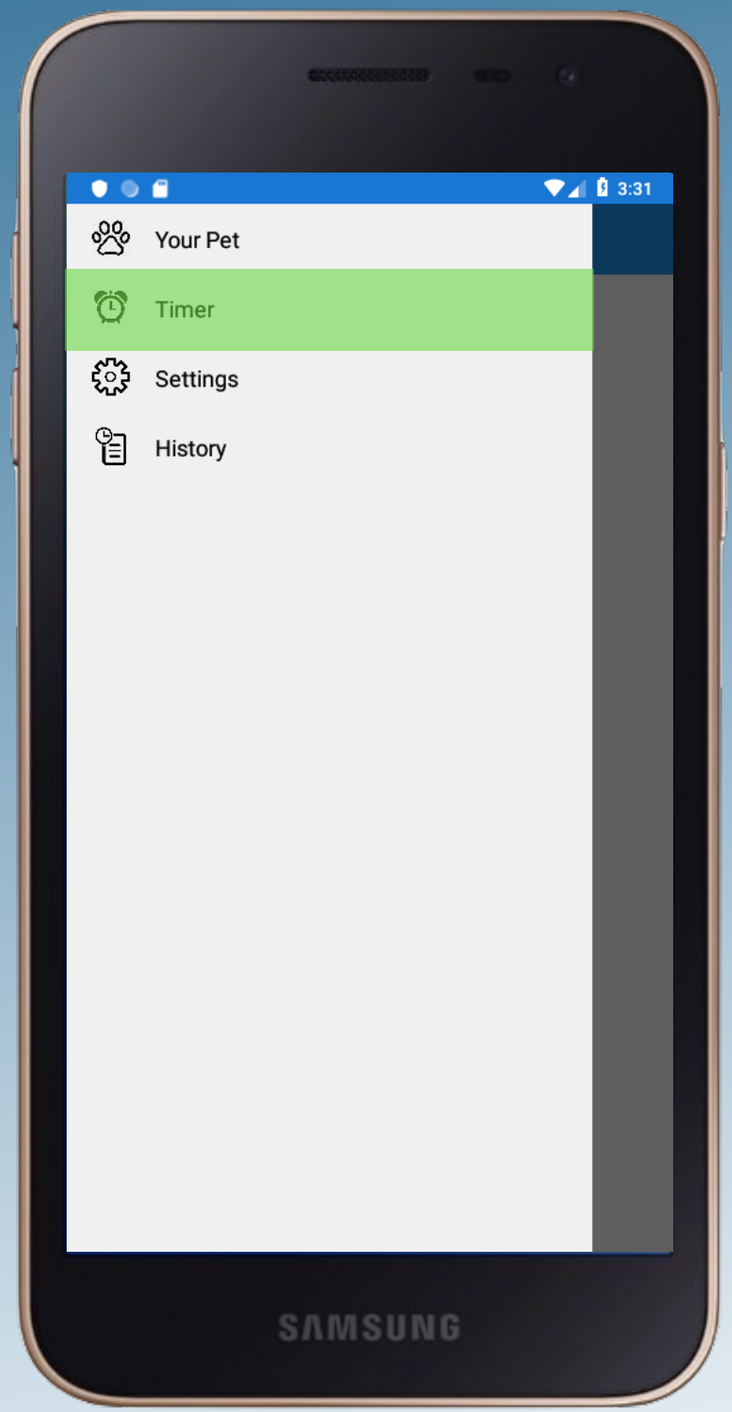
IOT Hub service

- Sending HeartBit message every 1 second



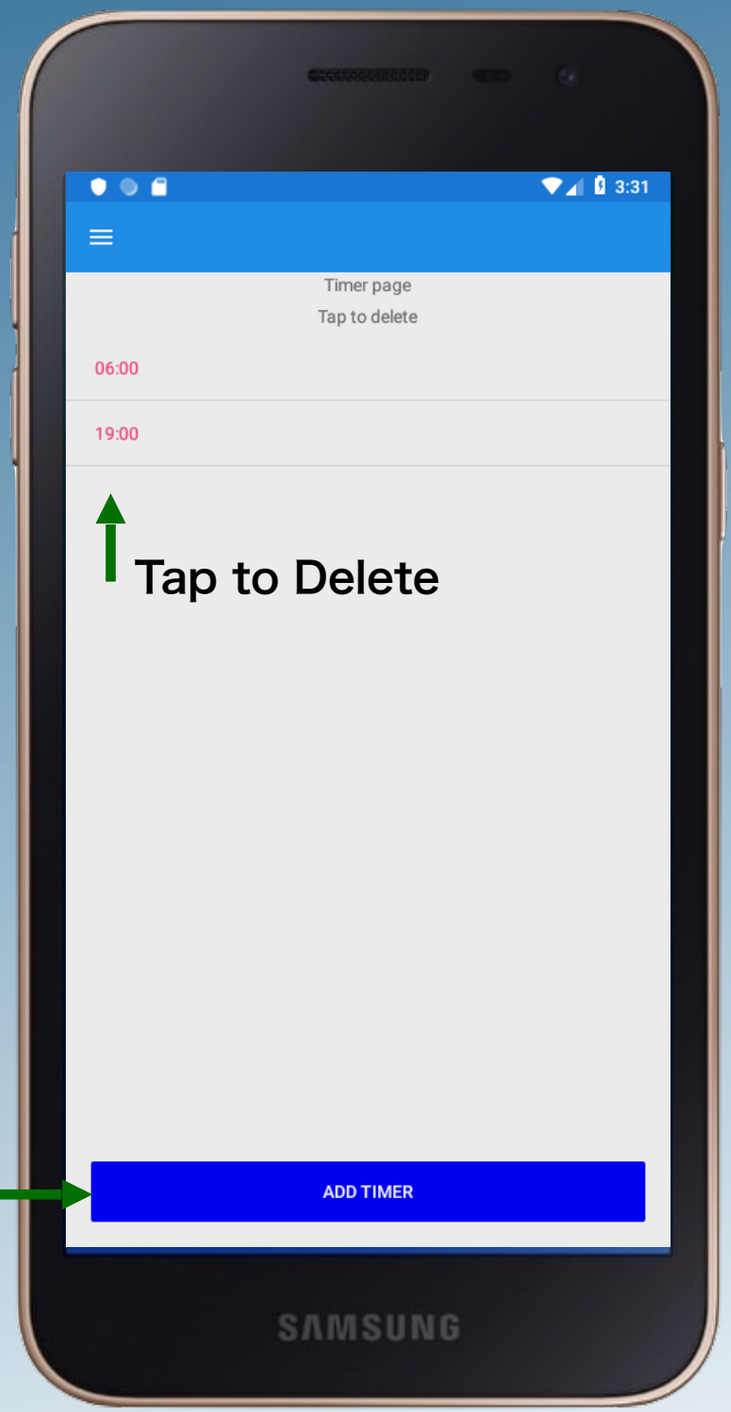
The Side-Menu

Enables Navigating between the App's different pages



The Timer Page

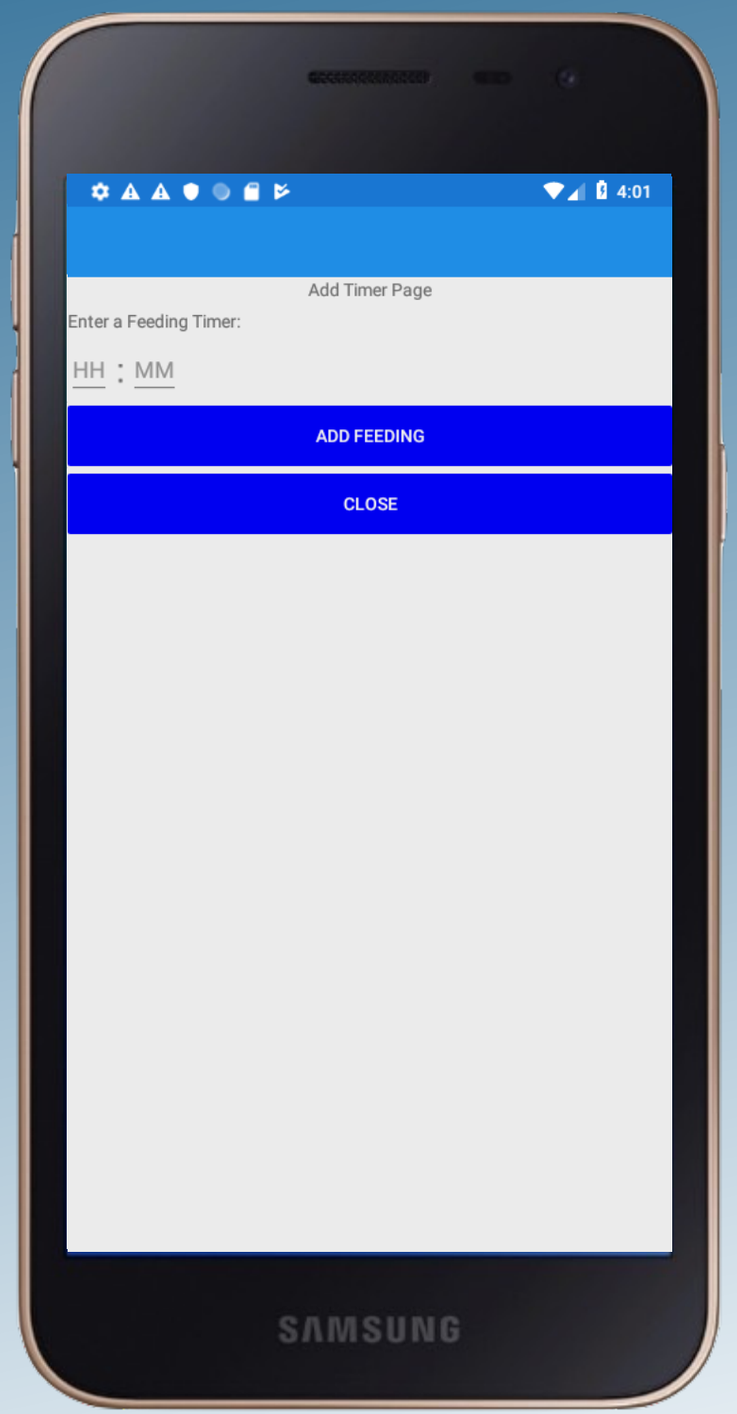
- Contains a list of all the scheduled daily feed timings that already exists
- Add Timer button



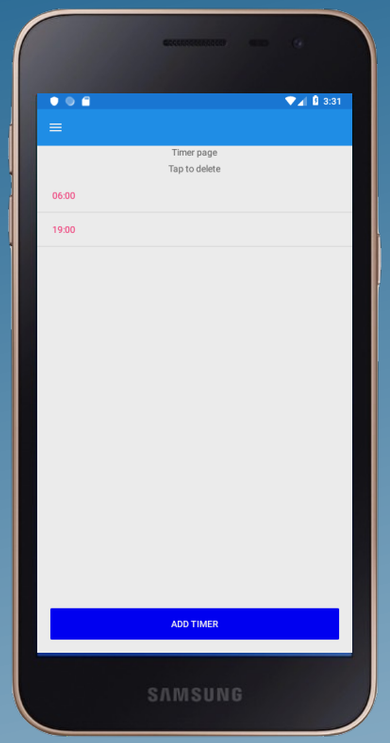
ADD Timer Button

Add Timer Page

Enter a daily-feeding Timer for any minute of the day



The Timer Page



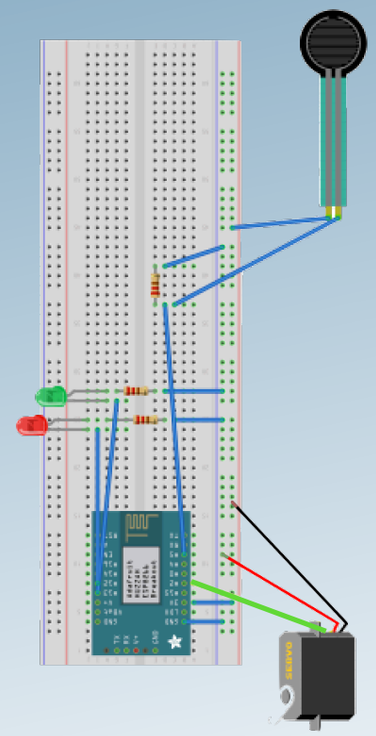
a Timer added in the app is stored in Azure DB Table

Feeding Times Table

Device Code (Partition Key)	Feed Timer
PetFeederC100LI	06:00
PetFeederC100LI	19:00

Feed by Time
Time-Triggered Azure Function

Activates Feeding in Device



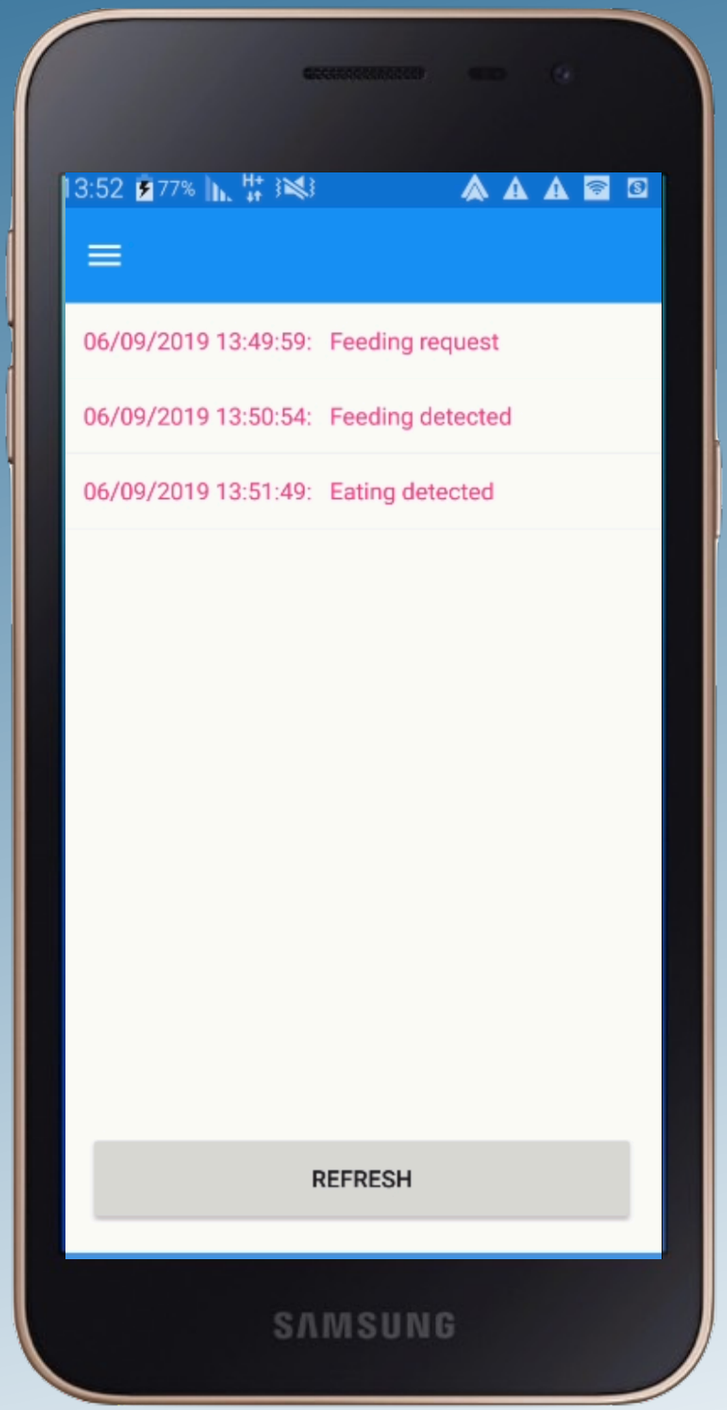
The History Page

Contains a list of all the Feeding activity that happened:

<u>Feeding request</u> - The FEED Button was pressed / a Feed Timer went on

<u>Feeding detected</u> - The Device's force sensor detects that the food was poured to the plate successfully

<u>Eating detected</u> - The Device's force sensor detects that the food had been eaten



The History Page

First stage: Collecting data

Message Reader
IoTHubTrigger Function

Write to Storage table

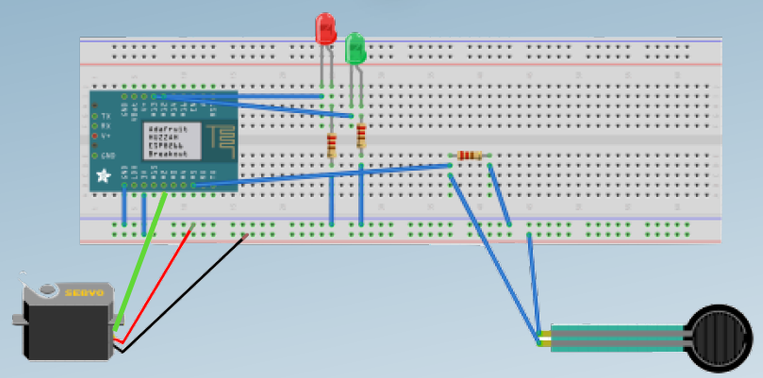
WeightsTable

Device Code (Partition Key)	MeasurementType	Weight
PetFeederC100LI	0	304
PetFeederC100LI	0	302
PetFeederC100LI	1	300
PetFeederC100LI	0	297
PetFeederC100LI	0	308

- Writes “DeviceId” to PartitionKey column
- Writes MeasurementType 0 which means this is only “unstable” sample
- Writes the received weight

IOT Hub service

- Sending message with current “Weight” returned by analogRead and “DeviceId” every 1 second



The History Page
Second stage: Analyzing data

MonitorWeight
TimerTrigger
Function

For each DeviceCode:
Reads data from WeightsTable

Updates WeightsTable

WeightsTable

Device Code (Partition Key)	MeasurementType	Weight
PetFeederC100LI	0	304
PetFeederC100LI	0	302
PetFeederC100LI	1	300
PetFeederC100LI	0	297
PetFeederC100LI	0	308

- Calculates new “Stable” weight from samples
- Cleans old “Unstable” weights from WeightsTable
- Decides what action occured: Eating/Feeding/Nothing
- AddToHistory(DeviceId,ActionType)

HistoryLogTable

Device Code (Partition Key)	User’s ID (Row Key)	ActionType
PetFeederC100LI	22313	FeedingDetected
PetFeederC100LI	22313	EatingDetected

The History Page
Pseudo code: MonitorWeight algorithm

MonitorWeight (Samples S)

- Triggered every 2 minutes
- X := “Old Stable Measurement” = S where MeasurementType = 1
- Y := “New Stable Measurement” = CommonOccurrence(S,&Percent)

- CommonOccurrence calculates which “unstable” weight is the most common. “unstable” weight is the weights from the table where MeasurementType = 0. The function also fills output percent that contains the percentage occurrence.

- Then, check if the sample meets some conditions. We check
  if (Size(S) > MIN_SAMPLES && Y.Percent > MIN_PERCENT)
  call UpdateEvent(X,Y)

- UpdateEvent(X,Y): Gets Old stable measurement “X” and New stable measurement “Y”. If (Abs(X-Y) > CHANGE_THRESHOLD) then:
  if (Y > X)
  AddToHistory(DeviceId,”FeedingDetected”)
  else
  AddToHistory(DeviceId,”EatingDetected”)

- Remove all samples where MeasurementType = 0
- Update “WeightsTable” where MeasurementType =1 set Weight = Y

·After some data analysis we used MIN_SAMPLES=60, MIN_PERCENT=20, CHANGE_THRESHOLD=10

The Settings Page

- Update pet's name
- Remove Device to start over