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.
Hardware

Our Board:

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

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

Control LEDs

FlexiForce Force Sensor
- Connected to the food platter and sends the user the current weight.
- Enables verifying and monitoring the feeding process
What its made of?

- Feeding bottle
- ESP8266 Board
- Plastic box
- Old drawer
- Pressure Sensor
- 3D Printed cap attached to Servo Motor
Azure Web Services

- Connected to the Devices and enables communication with them

Azure Functions

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

Azure Tables

- Users Table
- Devices Table
- Feeding Times Table
- History Log Table
- Weights Table
**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
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
- 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
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
Enter a daily-feeding Timer for any minute of the day
The Timer Page

Feeding Times Table

<table>
<thead>
<tr>
<th>Device Code (Partition Key)</th>
<th>Feed Timer</th>
</tr>
</thead>
<tbody>
<tr>
<td>PetFeederC100LI</td>
<td>06:00</td>
</tr>
<tr>
<td>PetFeederC100LI</td>
<td>19:00</td>
</tr>
</tbody>
</table>

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

Feed by Time
Time-Triggered Azure Function

Activates Feeding in Device
Contains a list of all the Feeding activity that happened:

**Feeding request** - The FEED Button was pressed / a Feed Timer went on

**Feeding detected** - The Device’s force sensor detects that the food was poured to the plate successfully

**Eating detected** - The Device’s force sensor detects that the food had been eaten
The History Page
First stage: Collecting data

<table>
<thead>
<tr>
<th>Device Code (Partition Key)</th>
<th>MeasurementType</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>PetFeederC100LI</td>
<td>0</td>
<td>304</td>
</tr>
<tr>
<td>PetFeederC100LI</td>
<td>0</td>
<td>302</td>
</tr>
<tr>
<td>PetFeederC100LI</td>
<td>1</td>
<td>300</td>
</tr>
<tr>
<td>PetFeederC100LI</td>
<td>0</td>
<td>297</td>
</tr>
<tr>
<td>PetFeederC100LI</td>
<td>0</td>
<td>308</td>
</tr>
</tbody>
</table>

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

IOT Hub service
Message Reader
IoTHubTrigger Function
Write to Storage table

- Sending message with current “Weight” returned by analogRead and “DevicId” every 1 second
The History Page
Second stage: Analyzing data

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

<table>
<thead>
<tr>
<th>Device Code</th>
<th>MeasurementType</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>PetFeederC100LI</td>
<td>0</td>
<td>304</td>
</tr>
<tr>
<td>PetFeederC100LI</td>
<td>0</td>
<td>302</td>
</tr>
<tr>
<td>PetFeederC100LI</td>
<td>1</td>
<td>300</td>
</tr>
<tr>
<td>PetFeederC100LI</td>
<td>0</td>
<td>297</td>
</tr>
<tr>
<td>PetFeederC100LI</td>
<td>0</td>
<td>308</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Device Code</th>
<th>User's ID</th>
<th>ActionType</th>
</tr>
</thead>
<tbody>
<tr>
<td>PetFeederC100LI</td>
<td>22313</td>
<td>FeedingDetected</td>
</tr>
<tr>
<td>PetFeederC100LI</td>
<td>22313</td>
<td>EatingDetected</td>
</tr>
</tbody>
</table>
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(Deviceld,"FeedingDetected")
  else
    AddToHistory(Deviceld,"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