

# Smart Outdoor Workout

Naseem S, Fares Sh, Milad B, Samih T

## Summary:

Our project basically focuses on bringing the street workout to be driven by technology to easily measure and monitor the workout sessions, and letting people keep track of their progress.

## Components:

1. Huzzah Feather – *ESP8266*:



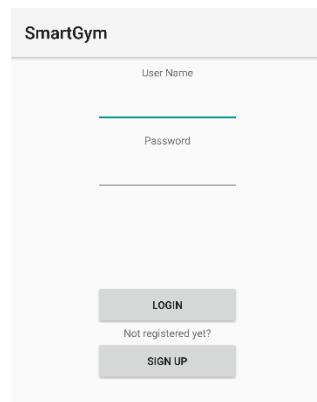
2. Distance sensor – *TFMini LiDAR*:



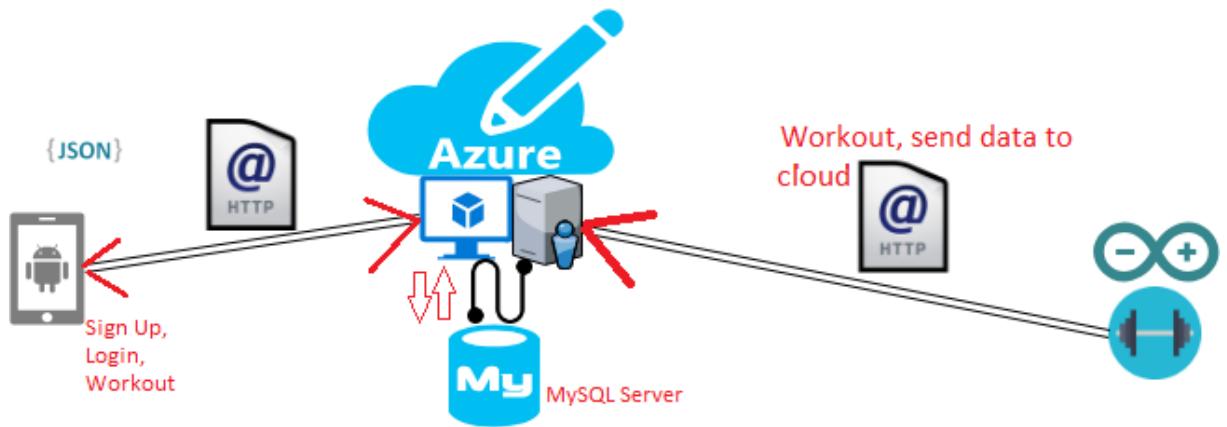
3. A Virtual Machine in Azure that runs a server:

	<b>NaseemVM</b>	Virtual machine
	<b>naseemgroupdiag696</b>	Storage account
	<b>NaseemGroup-vnet</b>	Virtual network
	<b>NaseemVM_OsDisk_1_eb6adcc...</b>	Disk
	<b>naseemvm855</b>	Network interface
	<b>NaseemVM-ip</b>	Public IP address
	<b>NaseemVM-nsg</b>	Network security gro...

4. Android application:



## How it all works:



Once the user signs up:

SignUp

First Name	<input type="text"/>
Last Name	<input type="text"/>
Username	<input type="text"/>
Password	<input type="password"/>
Age	<input type="text" value="30"/>
	<input type="text" value="29"/>
	<input type="text" value="31"/>
Weight (in KGs)	<input type="text" value="70"/>
	<input type="text" value="69"/>
	<input type="text" value="71"/>

His data is sent to the server by a HTTP request.

So, he can sign in, and immediately start training:

SmartGym

Hello faresss

Time elapsed 0:04:161

Sets 0

Repeats 0

Total Repeats 0

**STOP TRAINING**

HISTORY

LOG OUT

Once the user presses ‘start training’, the server starts getting data from the sensor which is placed on the training machine through HTTP requests, and saves them into the database.

Then he can monitor his work; number of repeats, number of sets, etc.

Sets	0	(The smart app automatically recognizes when a new set has begun).
Repeats	0	
Total Repeats	0	

And we do this from the app by requesting data from the server while training.

The user can even check his history of previous trainings after finishing his current one. This is left for you, the reader, to discover by yourself!

### Arduino:

The **TFMini** is a ToF (Time of Flight) LiDAR sensor capable of measuring the distance to an object as close as 30cm and as far as 12 meters.

The sensor works by sending a modulated near-infrared light out. The light that is reflected from the object returns to the sensor's receiver.

The distance between the two can be converted using the sensor by calculating the time and phase difference.

Pinouts and hookup:

3.3V Arduino	TF mini lidar
Software serial RX(13)	UART_TX (3.3V) - Green cable
Software Serial TX(15)	UART_RX (3.3V) - White cable
USB	5V - Red Cable
GND	GND - Black Cable

The sensor is placed on the machine and it calculates the distance from the floor.

- One problem was the sensitivity of the sensor:

For example, when the trainer makes a very little move it sends another data update (HTTP to server), because it's sensitive.

But in this case, we needed to make it like it's stopped in order to make it easier for us to detect when he's moving up or down.

The solution was determining a minimum value (cm's) to the difference between every two moves, and it's determined by observing the data.

- Another problem was the big amount of HTTP requests from the sensor to the server.

So, we minimized that amount by sending when it's only moving up or down (when he's stopped we don't send).

\*Note: We wanted to use both **TFMini Lidar** and another acceleration sensor called MMAQ8542:

After initializing the acceleration sensor and making it work properly with the **TFMini** simultaneously, we discovered that there's no need for it, and that is because we can compute the acceleration in each point by having the speed of the previous point and the time interval between the two points.

So, we decided to use only the **TFMini** instead and the reason is more appropriate and accurate data.

### Cloud:

In the cloud we have a server that is deployed on an Ubuntu virtual machine on Azure.

The server listens to HTTP requests (**REST APIs**) both from the Android application and the Arduino.

The server also contains a **MySQL** [database](#), which is concurrently updated upon receiving requests from the Arduino and the app.

An **Apache** server is deployed on the VM that listens to HTTP requests. Apache passes the requests via WSGI (Web Server Gateway Interface) to **Django** framework. The Django framework has a URL pattern matcher that maps every requested URL to the relevant Python function.

The server-side is written in **Python**, which uses the **Django** web framework and **Apache**. We chose this framework due to its ease of use, and we can handle the requests exactly as we want it from within Python.

Obstacles and problems encountered:

- At first, we tried deploying the server via the Azure UI, but it was complicated because it was unclear how to use it and manage every little configuration you want, so we created a virtual machine on Azure, made it open to accept SSH connections, and connected to it via the program **MobaXterm**, and from there we configured the Django framework, the MySQL Database (permissions to it, building it etc.).
- The Django server kept shutting off automatically after machine suspension (“sleep mode”). So we decided to use **Apache** to receive the requests, which sends them to the Django that handles them.
- Another interesting problem we encountered, was how to implement the data visualizations from within Python and the database. We solved by finding a rich Python library called **Plotly** that has a very easy to use API that lets you draw charts from within Python on the fetched data from the database.

### Android Application:

The application is built in **Visual Studio**, in **C#**, and it basically works via HTTP requests to the server and getting responses back.

Problems and obstacles:

We knew nothing about Android development, especially in C#, we saw some tutorials and downloaded some templates and played around with it until we knew how to add components to the app (buttons, text boxes, ...). We wanted to make a sign-up/sign-in mechanism, a start/stop session functionality, and show an overview after a workout is done (visualization), and we achieved that by Googling a lot and thinking how to implement such mechanisms.

How to use the app is described above, and in the video: <https://youtu.be/S1ZTrXLvgZo>