Performance Engineering of Wireless IoT sensors

International Institute Of Information Technology, Hyderabad

Performance Engineering Project
Monsoon 2016

Shaleen Garg (201401069)

supervised by
Dr. Anil Gurijala

Contents

Introduction

Pervasive sensing using Wireless Sensor Networks are commonly found in many faces of modern life. This enables us the ability to collect and infer from environmental indicators. These environments can scale from a small room to large cities and even country wide depending on the use-case.
The rapid increase of the number of these kind of devices functioning in a communicating and actuating network across the world creates a huge pool of cheap, highly extensible and low power consuming devices formerly unknown to mankind now called Internet Of Things (IoT).
This project was inspired by the extraordinary steps taken by the Telangana government in using IoT sensors in micro-climates across the state in order to crub the fatalities faced during extreme summer months due to heat strokes.
It set foot to install small sensors, connected via wifi or cellular connections to deliver temperature and humidity readings of micro-climates to a local hub tagged with its location so-as to infer if there is a possibility of heat stroke in that area. If such a possibility exists, the local hub delivers this news to the local hospitals who will then stock the necessary equipment to effectively treat the potential victims of heat stroke. They also aim to send SMS to all the localites to take precautionary measures to avoid heat strokes.
Using IoT devices makes the solution very cost effective.

Problems Addressed

This project gives a study on the performace aspects of a possible arduino based solution using wifi module ESP8266 and arduino Nano with the DHT11 Temperature and Humidity sensor.
The proposed arduino Nano and ESP8266 based solution makes a platform for an array of sensors which can be attached to it (limited by the number of pins on the arduino) in order to sense and actuate in any environment.
The fundamental tradeoff faced during setting up such a system in the field is the tradeoff between the power-consumption and the frequency of data collection. If the system has unlimited power supply, then it doesnt matter much and the frequency of data collection can be adjusted as required, but if there is a limited power supply available to the system, then the backup time is hugely affected by the data collection frequency.
Hence, this project will address the problem and give a basis data to the field developer while designing the system.
As a side aim, I tried to keep the whole system under Rs.1500/- ($22).

System Architecture & Technologies Used

The system used in this project consists of 3 main components.

1. Arduino Nano
   The arduino nano is an open-source electronics prototyping platform which has an Atmel ATmega328 8-bit AVR microcontroller with components which help running arduino programs directly on it without the need of an external program burner. It interfaces with the outside world using its 14 General Purpose I/O (GPIO) pins.
   It needs an external 5V power supply which can be given through its USB-B port or the 5V and ground gpio pins on the board.
   One can load Arduino Nano with the a program using the on-board usb serial controller connected to a computer through the arduino IDE.
   This particular version of arduino over the hundreds of versions available was used because it is small and has same capabilities as a regular arduino. Size make the final package small and managable.

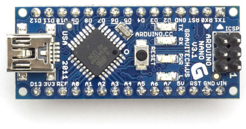

Figure 1: The Arduino Nano

2. ESP8266
   The ESP8266 is a coin sized wifi module with full TCP/IP stack and Micro Controller Unit. This module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands.
   It is designed to work with mobile devices, wearables and IoT applications. The power saving architecture features three modes of operation - active mode, sleep mode and deep sleep mode, thus allowing battery-powered designs to run longer.
   To facilitate lower power consumption, the module works on 3.3V power input with 3.3V logic levels. Hence to use it with the arduino, we have to use logic level convertors.

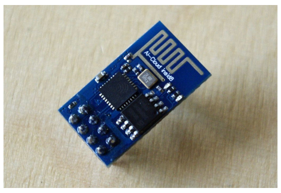

Figure 2: The ESP8266 WIFI Module

3. DHT11
   The DHT11 is a temperature and humidity sensor for the arduino. It needs a 5V power supply when being queried for the values, else it doesnt need any power.
   It supports temperature range of 0-50°C±2°C and humidity range of 20-90% RH±5% RH at maximum speed of one request per second.

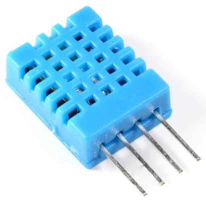

Figure 3: The DHT11 Temperature & Humidity Sensor

Lab Setup Configurations

For ease of developing the system for testing puropses, all the components were connected and tested using a breadboard; No hardwiring was done.
For the purpose of getting data wirelessly from the setup to a hub, a web2py server was configured.
sleep() command in arduino was used to wait for the specified amount of time before sending the data.
The setup was first run without any power saving features.
It was observed that the power consumption did not change with the change in the frequency of data collection.
This goes to say that if the setup has mains connection, then frequency of data can be adjusted to the

demand.
After this, the code was modified with deepsleep mode for ESP8266 being enabled ("AT+GSLP"). This reduced the power consumption significantly and change in frequency reflected a change in the power consumption.

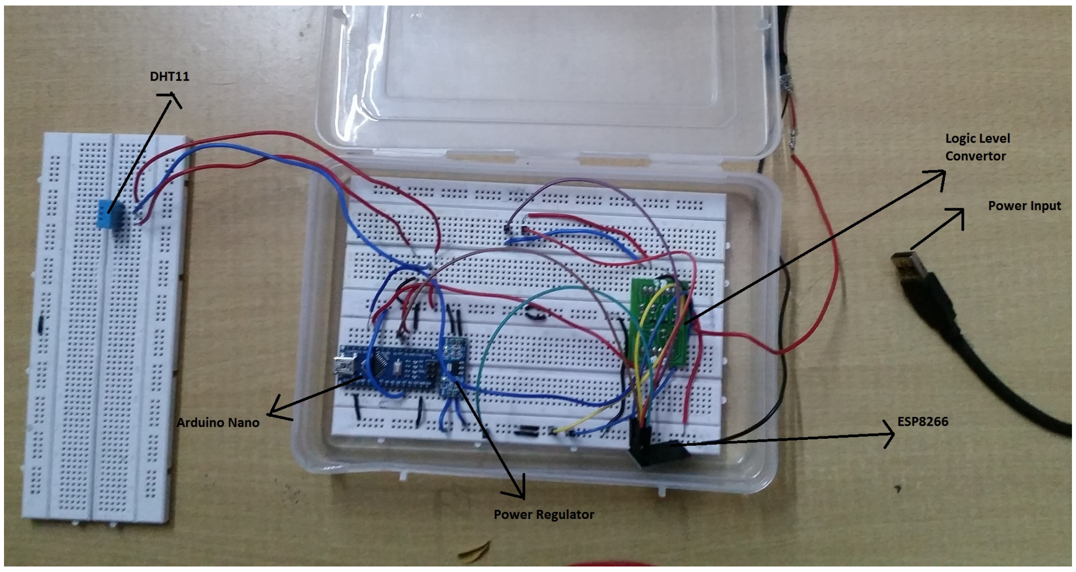

Figure 4: The Final Setup

Test Processes & Results

DHT11

The readings taken and projected in a graph showed significant amount of fluctuations, which goes to show that the readings given out by the sensor are unreliable and can be used only for speculative purposes. The mean values of different time periods show a relative change (Not completely arbitrary), but individual values may be missleading.

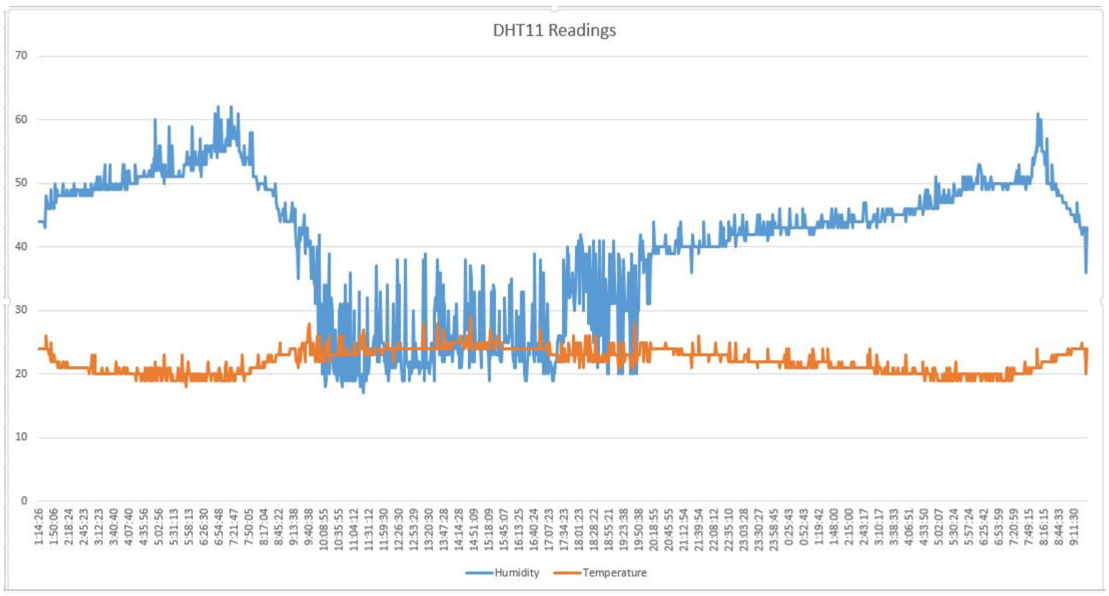

Figure 5: Highlighting the fluctuations in readings

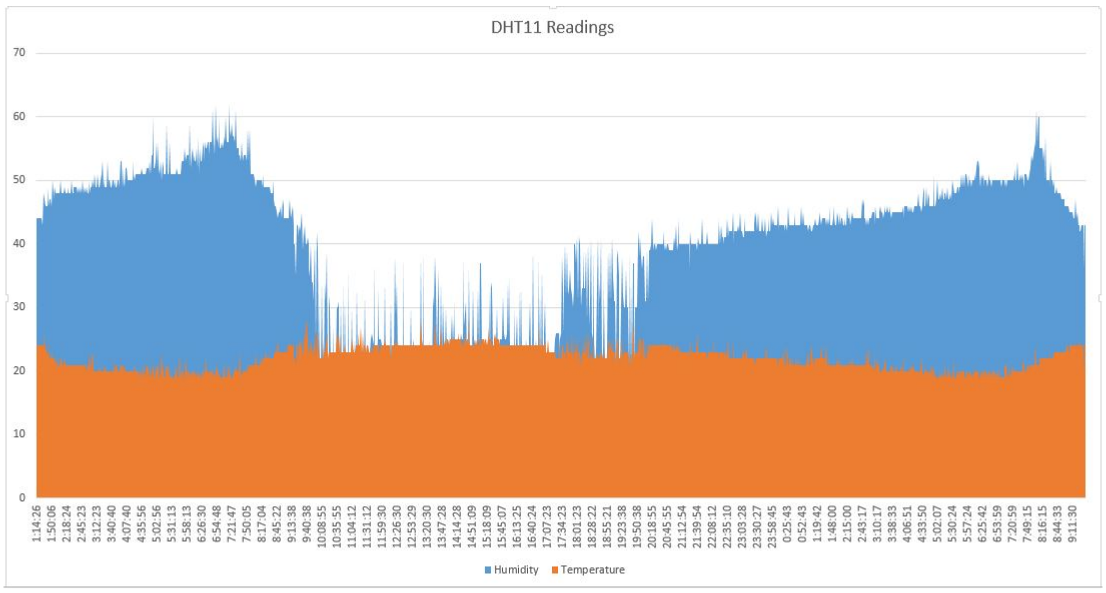

Figure 6: Showing the change in average trends with the time of day

ESP8266 and Arduino

It is known that message size from the setup will affect the power consumption of the system; In our case, the message size remains constant. Hence all the readings can be assumed to have all other factors constant.
To see the changes in the power consumption, time between data transfers has been changed in 5 min increments from 5min - 30 mins. Each change was recorded for 24 hours to get a stable reading.
For the experiment, 4 standard AA zinc-carbon batteries were simulated to plot graphs of the backup time if the respective data collection frequencies are used.
Four, because each AA battery is 1.5V; our setup needs around 5V atleast to remain functional, $4 * 1.5V = 6.0V$.
This will give a fair idea of how much battery backup should be provided to the system in order to have a stipulated backup time.
In the tables and graphs, it is assumed that each AA battery has around 2400mah of power i.e. 9600mah in 4 AA batteries.

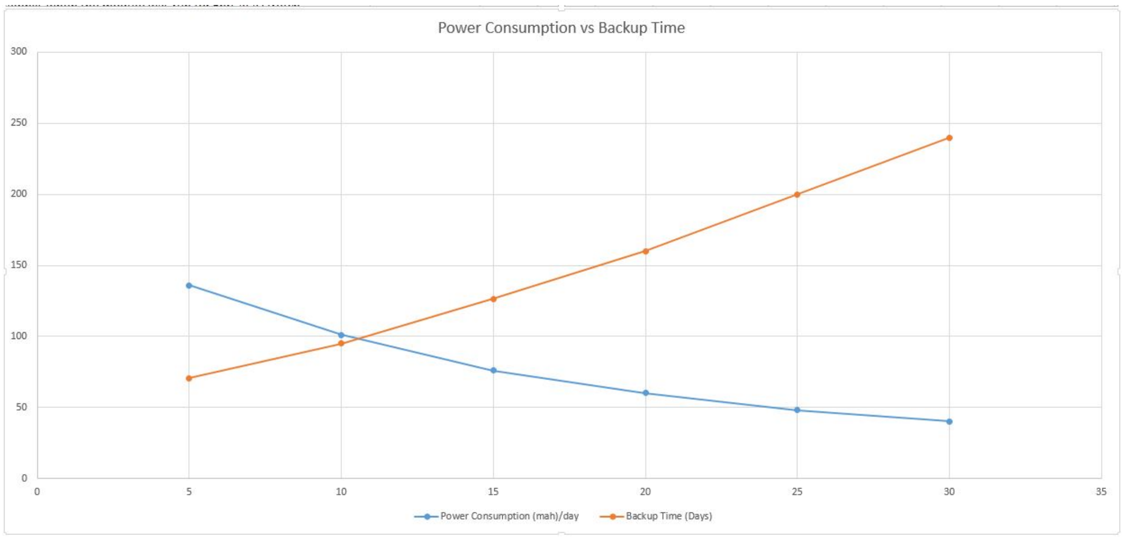

Figure 7: Power Consumption vs Backup Time with 9600mah

Sleep Time (mins)	Power Consumption (mah/day)	Power Consumption (mah/hr)
5	136	5.67
10	101	4.21
15	76	3.16
20	60	2.5
25	48	2
30	40	1.67

Table 1: Power Consumption and Sleep Time Table

Discussion

Now that we have a graph between Power Consumption and Backup Time, given the battery size, what is the best interval between data collection?
From the reliable weather data trends, it can be reliably deduced that temperatures of a micro-climate do not change very rapidly in one hour slots, hence it would'nt be a loss if we take readings each hour, but in such a scenario, we loose the power of predicting heat strokes since they come rapidly and high temperatures may have already reached before being captured by the setup. This one-hour reading will be ideal if one wants to just take the day's highest and lowest temperatures but given the erroneous nature of the DHT11 sensor, that may give us false readings.
One way to get the best results with reduced power consumption is to change the frequency of readings dynamically with the time of day, i.e. between 1200 and 1500 hrs, one can expect the highest temperatures of the day, hence increase the frequency of data collection in that period of time; likewise between 0100 and 0300 hrs, lowest temperatures of the day can be expected, hence increasing the frequency to have the lowest temperature of the day. All other time periods of the day are expected not to have the highest or the lowest temperatures, hence dont need to have high data collection rates.

This project was a completely new but interesting domain for me. Working in an unknown territory gave rise to some very fundamental challanges while setting up thesystem. I had never programmed in an arduino, knew very little about the way arduino works. These were just the beginning of the whole mountain of problems coming up next. Course in Electronic circuits had given me enough knowledge to start with the breadboard , gpio pins, how not blow up the ICs.
It was like a journey to me, selecting the components, reading all the data sheets of each component so-as to know their SLAs. Some components that were bought were not at all needed for the completion of the projection but were bought due to sheer lack of understanding of the components. Also many components were bought as I progressively understood their need for the project.
I used online learning sites like Coursera, Youtube and other written documents available on the internet to understand and build the system from scratch.
When first testing my ESP8266, I did not know that it needed significant power (300 - 400 mA) to start up and send data throught wifi. I accidently burned my first wifi chip due to short-circuit. I also burned my fingers while soldering the logic level convertor to its gpio pins (never used a solder before).
This project was a first for many things. I learned how to plan and structure the placements on the breadboard before actually pushing anything on it so that debugging is simpler and cleaner after everything is in place.
One of the biggest problems faced during the course of the project was to come-up with a solution to find the power consumption of the system while it is running on the external power supply. This was overcome using a usb power consumption monitor. To make use of the monitor, I stripped a usb cable and connected the bare wire ends to the 5V and GND bus of the bread board. Since all the components were connected to it, it worked seemlessly, giving me accurate power consumption readings.
It was the mental support and exceptional cool of my professor, Dr. Anil Gurijala, that kept me from abandoning the project when I was unable to make progress in it.

Conclusion

The initial side aim of keeping the whole project under Rs.1500/- was successful. This was still because I bought just one component at a time. One in large scale production, this cost will further go down. This means that each such system will cost under $22! a remarkable achievement in contrast to costly commercial solutions which cost 100 of dollars.

This project is a remarkable feat in setting up a power consumption benchmark for an arduino based wireless remote sensor with limited power resourses. It can be used by researchers and enthusiasts across the field in getting different kinds of data for different use cases; may it be sensing the activity patterns of an animal in the forest, or sensing water levels of oceans in different geographical locations, or monitoring the activities in the cities et cetera.
This project gives those people a rough estimate of what behaviour can they expect from their finished product which is very hard to predict before hand.