Design and Analysis of Low Power Sensor Nodes

[1]D'Costa Brenner Branco Rosario
[1] M.E. Student, Department of Electronics and Telecommunication Engineering, Goa College of Engineering, Ponda Goa

Abstract*— There are many sensor nodes in the environment like soil moisture sensors, temperature sensors, pollution sensors, etc. The operation of these sensors is controlled by the microcontrollers that they are interfaced with. The main limitation of these sensor nodes in the environment is their battery life. If the battery life is just a few weeks, then the battery has to be replaced several times a year. The approach here is to minimize the current consumption of sensor nodes like the esp8266 Wi-Fi module that will be interfaced with sensors so that it is capable of working under very low power. And the battery supplying power must be functional for at least half a year since it is convenient to replace the battery of a sensor node once a year rather than having to replace it every few weeks. The methods that need to be adopted so as to reduce power consumption of sensor nodes to achieve power efficient employment of Internet of Things (IoT) is discussed in this paper.

IndexTerms—*esp8266,IoT,sensors,WiFi*

I. INTRODUCTION

The number of sensor nodes in the environment is increasing at a rapid rate daily in order to facilitate the Internet of Things (IoT). Such sensors are used in data analysis of physical phenomena like changes in the weather, soil moisture and humidity level, pollution indication and much more. These sensors need to be deployed in the environment and they also need to work efficiently for a long period of time without draining the battery completely. In order to achieve this, the sensor node needs to be programmed and designed in such a way that power is used sparingly since battery power is always limited. The world is going to witness an exponential rise in the devices connected to the Internet to enable the Internet of things in the coming years. Hence the requirement of battery-operated low power sensor nodes is also going to be crucial. The main sources of power consumption are transmitting or resaving data, Processing query requests, forwarding queries and data to the neighbors which together constitute useful power consumption. Transmitting with high power in decibels per milliwatt to devices that are placed nearby, idle listening to a wireless channel waiting for possible traffic and transmitting beacon frames to detect the presence of wireless devices in the vicinity constitute wasteful power consumption. This paper discusses how the sources of wasteful power consumption were minimized, restricted and regulated. An approach was proposed for the ultra-low power implementation of control-oriented application tasks of a WSN (Wireless Sensor Network) application [1]. Their approach was based on power-gated hardware tasks that are implemented as specialized hardware blocks. The synthesis results for the hardware tasks of the case study application graph show that, compared with the MSP430 micro-controller and under a very conservative assumption, power reductions by two orders of magnitude are possible. Another related work [2] proposed cost-effective and self-powered networking devices for an Internet-accessible wireless environment monitoring system. The devices could harvest energy from various sources including solar and ambient RF radiation, and wireless power transmission (WPT). An electronic system [3] was devised to investigate the amount of current drawn by sensor node from battery in operating condition. In this work, an electronic system based on a dedicated PCB to visualize node current consumption usage and charge extracted from the battery during node operating states was presented. They selected benchmarks that represent usual tasks in WSN applications and node current consumption was experimentally analyzed. In [4], the problem of power consumption reduction in Wireless Sensor Networks used in gas sensing applications was discussed and the power consumption of a gas sensor node was reduced to 85.68 mW. The results obtained in this paper achieve reduction in power consumption well below this level by duty cycling the sensor node under consideration (ESP8266 Wi-Fi module) and also by executing a few tricks in code that reduces the average time that the radio stays on for.

Methods of reducing power consumption of sensor nodes is discussed in this paper, initially discussing the basic operation in section II. Few additional tricks, work arounds and methods of optimization are discussed in section V. Section VI discusses the methodology and flow diagram. Battery life is also estimated through calculations and is depicted in section VII.

II. FUNCTION OF A SENSOR NODE

A. Modus Operandi

A Wi-Fi signal transmission/reception node (ESP8266 Wi-Fi module) was used in this project. and the data sensed by the sensor was processed and transferred to the cloud server as depicted in Fig.1. A hardware circuit was designed for the sensor. The microcontroller (ESP8266) was programmed depending on the behavior of Wi-Fi and the sensor was activated accordingly. The battery serves as the main power source for the system. The sensors are interfaced with the microcontroller (ESP8266). The microcontroller was programmed with the help of a computer. The microcontroller should have Network facility in order to communicate with

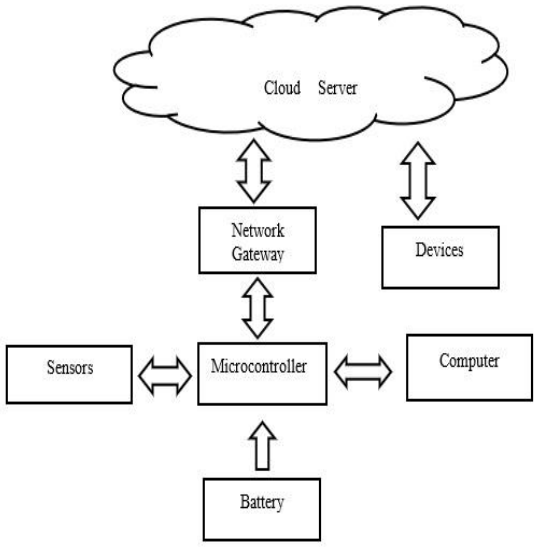

Figure 1. Functional Block Diagram

the network gateway which can be an access point, a router or even Lora WAN. ESP8266 is one such microcontroller that contains network functionality through its robust TCP/IP (Transmission Control Protocol / Internet Protocol) stack. It can be inferred from Fig. 1. that the sensors, the computer and the network gateway exchange data with the microcontroller.

B. ESP8266 Wi-Fi Module

The ESP8266 Wi-Fi module can function as a microcontroller. The ESP8266 board currently uses only 20% of its capability. The board is preprogrammed to operate with an average current of 80 mA. However, current measurements of lower orders of magnitude can also be achieved if the board is programmed carefully so as to consume power sparingly. Fig. 2 displays an ESP8266 NodeMCU module that contains several external components to regulate power to the board.

The ESP8266 has the following features:

- General-purpose input/output (16 GPIO)
- Inter-Integrated Circuit (I^2C)
- IS interfaces with DMA (sharing pins with GPIO)
- UART transmit-only UART on GPIO2)
- Pulse-width modulation
- Low-power highly-integrated Wi-Fi solution

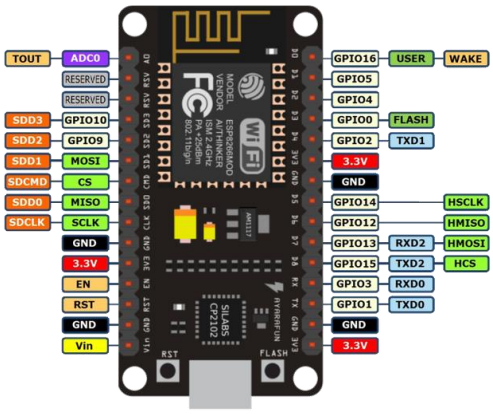

Figure 2. ESP8266 NodeMCU Module

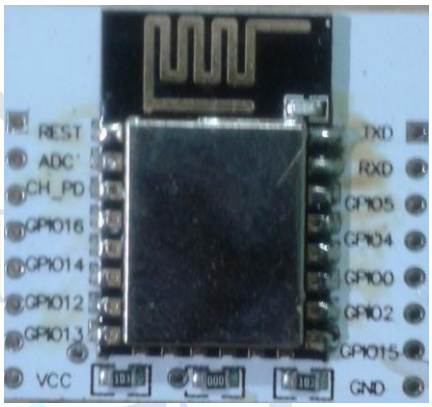

Figure 3. ESP8266 Generic Module

The ESP8266 Generic module displayed in Fig. 3 uses bare minimum components to power itself from a battery. It is because of this reason that it is capable of consuming current in the lower range of micro-amperes in deep-sleep mode as compared to milliampere power consumption of the NodeMCU module in the same deep-sleep mode.

III. RTC MEMORY

The RTC memory in the ESP8266 Wi-Fi module is responsible for waking the ESP8266 from deep-sleep mode after a predefined time interval has passed. This means that even though the ESP8266 and most of its on board components are not operational in deep-sleep mode, the RTC memory does remain on and once the time interval defined for deep-sleep has passed, the GPIO 16 pin on the ESP8266 sends a low signal to the RESET pin of the ESP8266 and hence resets the module so that the entire algorithm can be executed again. The RTC memory of the ESP8266 reserves 256 bytes for the system (system data) and 512 bytes for the user (user data). So, in total, the RTC memory can store 768 bytes of data.

IV. THINGSPEAK CLOUD SERVER

A cloud server named ThingSpeak which is open source was used for uploading sensor data. The http protocol was used to transmit data sensed by sensors to this cloud server. The variation in the values measured by the sensors can be monitored by the user over a period of time. The user only needs to create an account with the ThingSpeak service and then create his/ her channel. A channel id is then issued to the user to distinguish the channel from other channels. Within the channel, the user can create a maximum of 8 fields that are associated with every sensor that is interfaced

with the microcontroller. The values measured by the sensors are stored in these respective fields.

V. POWER SAVING TECHNIQUES

A few smart power saving techniques are discussed in the subsequent subsections *A*, *B* and *C*.

A. Range Based Signal Transmission

From the Table 1, it can be observed that the transmission power of the Wi-Fi module is between 14dBm and 20dBm but the receiver sensitivity is as low as -91 dBm. So, when the receiver is capable of receiving at such low power, then the wifi signal transmission power of the module can be made less depending on the distance of the gateway or standard wifi router from the wifi module. For example, if the wifi module i. e. the ESP8266 is in one room where a standard wifi router is also placed. If Received Signal Strength (RSSI) is strong from the router within that range, the ESP8266 exchanges data with the standard wifi router and while communicating with the router, it transmits some of its wifi signal power outside the room as well which is undesirable since, power is being wasted in this manner. To solve this issue, the ESP8266 needs to be programmed in such a manner that the power of the signal going to the antenna from the IC will be reduced. In this way if the power of the signal that goes to the antenna that transmits is reduced, then communication between ESP8266 and the standard wifi router can be achieved by using the wifi power sparingly only within that range and the RF signal strength won't be strong beyond that range.

Table I. ESP8266 EMISSIONS

Physical Mode	Transmission Power	Reception Power
802.11b	+ 20 dBm	- 91 dBm
802.11g	+ 17 dBm	- 75 dBm
802.11n	+ 14 dBm	- 72 dBm

B. Making the sensor wake up every few hours

Naturally occurring phenomena is sensed by the sensors interfaced with the ESP8266 wifi module, and this Wi-Fi module transmits this data measured by sensors to the internet via WiFi every 100 ms by default. When the physical conditions that are being sensed by the sensor are instantaneous and not abrupt as in temperature sensing applications, still the sensor transmits the data which is actually not required. Power is being wasted there. The ESP8266 board can be programmed to wakeup every 2-hrs to sense physical data and remain in sleep mode the remaining time. The ESP8266 wifi module also has a deep sleep mode. In order to reduce the current consumption to a few micro-amperes when the board is in this mode, the ESP8266 has to be duty-cycled carefully so that the average current over the entire cycle measures low as shown in Fig. 4

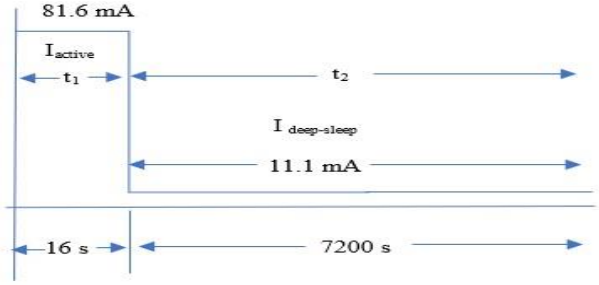

Figure 4. Duty Cycling ESP8266

Let us assume that the battery supplying the ESP8266 NodeMCU module has a capacity C of 500 mA-hr, then:

$$C = 500 \text{ mA-hr}$$
$$= 500\text{mA} * 3600\text{secs}$$
$$= 1800000 \text{ mA-secs} \quad (1)$$

When the ESP8266 NodeMCU WiFi module is transmitting data to the internet via Wi-Fi, let's assume that the active state current (I_{active}) is measured as 81.6 mA

$$I_{active} = 81.6 \text{ mA} \quad (2)$$

While the ESP8266 NodeMCU WiFi module is in deep-sleep mode, let's assume that the deep-sleep current ($I_{deepsleep}$) measures as 11.1 mA.

$$I_{deep\text{-}sleep} = 11.1 \text{ mA}$$

1 cycle of current measurement contains T secs

$$T = t_1 + t_2 \quad (3)$$

$$T = 16 \text{ secs} + 7200 \text{ secs} = 7216 \text{ secs}$$

Hence, the average current consumed in 1 cycle is:

$$I_{average\ (mA.secs)} = \int_0^{t1} Iactive\ dt + \int_{t1}^{t1+t2} Ideep\text{-}sleep\ dt \quad (4)$$

$$= \int_0^{16} 81.6 \text{ mA } dt + \int_{16}^{16+7200} 11.1 \text{ mA } dt$$

$$I_{average\ (mA.secs)} = 81.6\ (16 - 0) + 11.1\ (7216 - 16)$$
$$= 81225.6 \text{ mA-secs}$$

The average current in mA $I_{average}$(mA) is given by:

$$I_{average} = \frac{Iaverage\ (mA\ secs)}{T} \quad (5)$$

$$I_{average} = \frac{81225.6\ (mA\ secs)}{7216\ secs}$$

$$I_{average} = \frac{81225.6\ (mA\ secs)}{7216\ secs}$$

$$I_{average} = 11.2563 \text{ mA}$$

If the estimated life-span of the sensor node is denoted by L_B

$$L_B = \frac{C}{Iaverage} \quad (6)$$

$$L_B = \frac{500 \text{ mA-hr}}{11.2563 \text{ mA}}$$

L_B = 44.419 hrs

= 1.85 days.

In this way, if we increase the sleep time of the ESP8266 NodeMCU module as well as ensure that the WiFi circuit of the module is turned off while the interfaced sensors sense natural phenomena, the average current over the entire cycle can be reduced even further.

C. Restricting the beacon frames emitted by ESP8266

The beacon frames that are emitted by the wifi module actually are to only show the host that it is connected or in physical terms to tell the host that it is alive and ready to talk or communicate. These frames are transmitted periodically (every 100ms) by access points (APs) and wifi modules to announce the presence of LAN (this is in case of hosts). After the host (standard wifi router) and the client (ESP8266 wifi module) are connected to each other over the wifi network, then the beacon frames are useless. Power is consumed due to the transmission of these beacon frames. Hence, the ESP8266 Wifi module should be programmed in such a way that the beacon frames are transmitted by it every 2hrs. In this way power can be reduced to a huge extent.

VI. SYSTEM OPERATION

In Fig. 5 the basic system operation is depicted. The entire system is powered by a battery and the output terminal of a Passive Infrared Sensor (PIR) is connected to the ESP8266 WiFi module's CHIP_ENABLE pin (CH_PD) as well as one of the GPIO pins of the ESP8266 so that the status of motion detected can be stored in a variable and sent to ThingSpeak via WiFi and the Internet. Let's call this node, Node 2. The other sensor node (Node 1) is also an ESP8266 WiFi module that is interfaced with an LM35 temperature sensor for sensing variations in ambient temperature. The PIR sensor's input terminal is directly powered by the battery. If motion is detected by the PIR sensor, then the output of the sensor goes high and hence turns the enable pin high on the ESP8266 WiFi module that is named Node 2. The module executes all the instructions that it was programmed to execute as long as the enable pin on it is HIGH. After the PIR sensor's output goes low, the enable pin is pulled low of the ESP8266 and hence, it cannot execute the instructions that are loaded onto it. This is done to save power. It is necessary to ensure that Wi-Fi is turned off during the initial phase of program execution so as to avoid the large current spike that results as the module tries to reconnect to an access point after it wakes up from deep-sleep. The Wi-Fi circuitry had to be turned off before the ESP8266 could activate the sensors that are interfaced with its GPIO pins. After the sensors finish

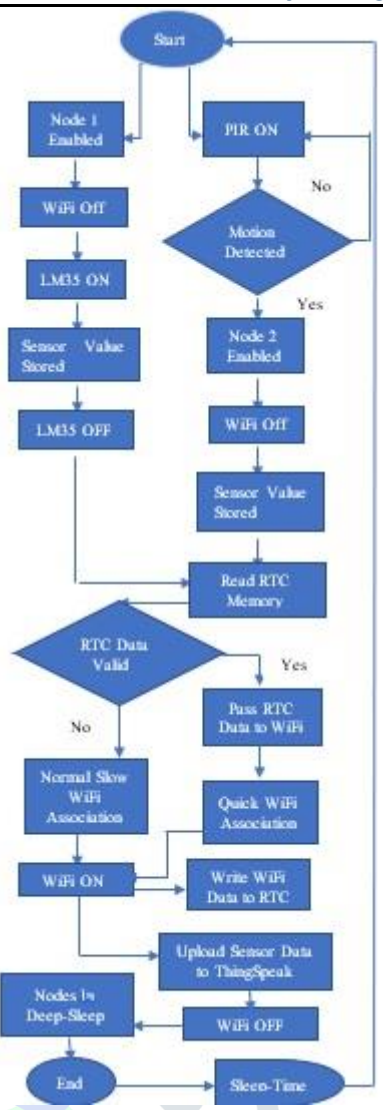

Figure 5. Schedule Algorithm

measuring natural phenomena, they are turned off by the ESP8266 throught the GPIO pins that they are interfaced with. Now, the WiFi circuitry in the board can be signalled to turn on. Before initiating an association with the WiFi access point or router, the ESP8266 checks whether the WiFi credentials of the access point it is supposed to connect to exists in the Real Time Clock (RTC) memory or no. If the Wi-Fi credentials do exist in the RTC memory, then the Esp8266 passes these values to the WiFi router or access point during the association period with WiFi andconnects to the Wi-Fi access point in a quick fashion which takes barely 2 seconds to complete. If the ESP8266 reads that the RTC data is invalid on the basis of a mismatch in the CRC32 (Cyclic Redundancy Check for 32 bits) or if it detects that the RTC memory contains no Wi-Fi credentials stored in it, then the ESP8266 tries a normal Wi-Fi association.

a) Normal Wi-Fi Association: In a normal Wi-Fi association, the ESP8266 device tries to make a connection with an access point. The access point and the ESP8266 module complete a complex handshake procedure, wherein, th access point or router issues the ESP8266 a unique IP Address via a process called Dynamic Host Configuration Protocol. Even though this process takes only 12 to 14 seconds to complete, in networking and wireless technology parlance this is considered to be quite a lot of time.

b) Quick Wi-Fi Association: In a quick Wi-Fi association, the procedure of initiating and completing a connection with the Wi-Fi access point is not as lengthy as in the case of a normal Wi-Fi connection. Firstly, the ESP8266 Wi-Fi module is configured with the help of code, to have a static IP address. Because of this, the

access point or router doesn't have to spend time in issuing the ESP8266 Wi-Fi module an IP address. 10 to 12 seconds are saved in this way. Hence, the time taken to make an association with the Wi-Fi router or access point takes around 4 to 6 seconds and the entire process from start of the algorithm right till the end when the ESP8266 goes into sleep mode takes only 6 to 8 seconds on average. In the context of wireless communication, time is energy.

Operating Mode	Current Consumption (mA)	Duration (secs)	Average Current $I_{average}$ (mA)	Expected Lifespan (Hrs)
Active	78 +- 0.62	2	10.939 Or 11.116	62.987 Or 62.972
DeepSleep	11 +- 0.088	7200		

c)Quicker Wi-Fi Connection with RTC: Performing cyclic redundancy check on the data stored in RTC memory is highly important in order to maintain integrity of data. In other words, CRC 32 is a protocol that ensures that the data received by RTC memory matches the data that was transmitted by the ESP8266. As depicted in Fig. 5, if the crc matches, then the RTC data is considered valid and hence passed on to Wi-Fi, else, the RTC memory is considered invalid or empty and a normal slow Wi-Fi connection is established with the wireless access point.

VII. RESULTS

As it is indicated from Fig. 6, the ESP8266 NodeMCU module consumes (78 +- 0.62) mA of current in active mode. The accuracy of the multimeter was considered which is ± (0.8%+2dgts) on the 200mV range. The battery used to power the ESP8266 NodeMCU board was a 9V eveready battery that had a current capacity (C) of 700 mA.hr. From Fig. 7 it is observed that this ESP8266 NodeMCU board consumes (11 +- 0.088) mA in deep-sleep mode. This amount of current is too high for low power applications. If we substitute the values for I_{active} and $I_{deep\text{-}sleep}$ in Equation (4),

$I_{average\ (mA.secs)}$ = 78798.72 mA.secs OR 80069.28 mA.secs

Substituting $I_{average\ (mA.secs)}$ in Equation (5) gives

$I_{average}$ = 10.9397 mA OR 11.1161 mA

Substituting $I_{average}$ in Equation (6) gives

L_B = 63.987 hrs OR 62.972 hrs

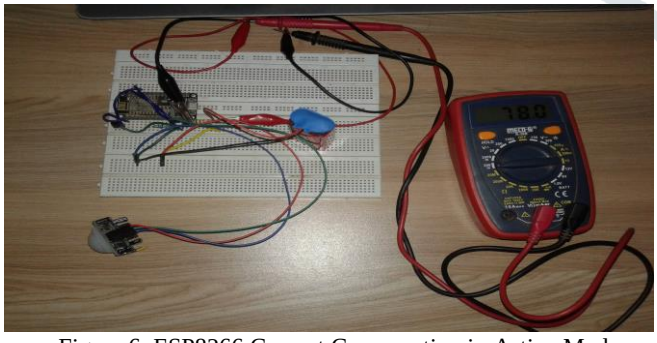

Figure 6. ESP8266 Current Consumption in Active Mode

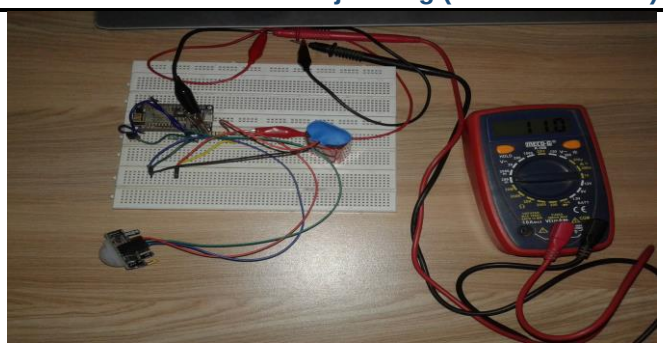

Figure 7. ESP8266 Current Consumption in Deep-Sleep Mode

Table II. ESP8266 NodeMCU Current Consumption

This means the battery will survive for only over a day if we use the ESP8266 NodeMCU module that consumes high current even in deep-sleep due to the additional components mounted on it like the AMS1117 voltage regulator and the CP2102 USB to UART integrated chip. Hence, the ESP8266 bare bones (Generic ESP8266 12E) that has none of these additional components mounted, was also tested to check if it draws lower current compared to ESP8266 NodeMCU module.

Figure 8. ESP8266 Generic Module Current Consumption in Deep-Sleep Mode

Table III. ESP8266 Generic Module Current Consumption

Operating Mode	Current Consumption	Duration (secs)	Average Current $I_{average}$	Expected Lifespan (months)
Active	78 (mA)	3	<132.21 (µA)	> 7.35
DeepSleep	< 100 (µA)	7200		

As observed from Fig. 8, the multimeter doesn't have enough resolution to measure the low current drawn by the ESP8266 12E Generic module. Since it shows 0.0 mV across the 1 Ω resistor, let's presume that the maximum current it draws in deep-sleep mode is 100 µA (since it doesn't read as 0.1mV on the multimeter which is equivalent to 100 µA). Substituting $I_{deep\text{-}sleep}$ = 100µA in Equation (4) gives a battery life of:

$$L_B = \frac{5294.61\ hrs}{24\ hrs} = 7.35\ months$$

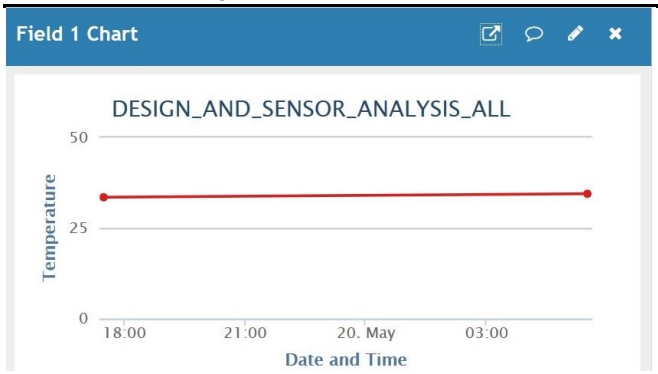

Figure 9. Temperature Data uploaded to ThingSpeak

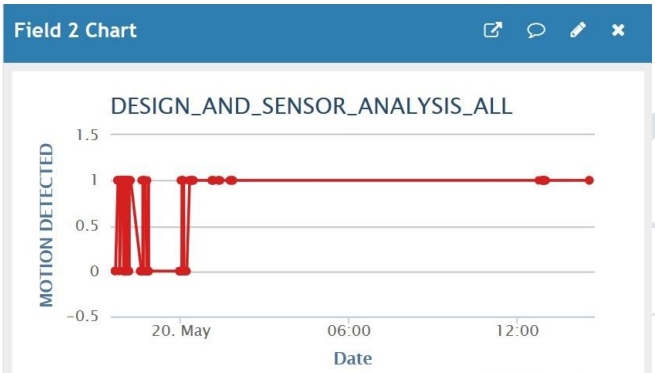

Figure 10. PIR status uploaded to ThingSpeak

ACKNOWLEDGMENT

The I am deeply thankful to my guide, Assistant Prof. Geeta Shet for allowing me to carry out the project under her supervision. She has supported me in times of difficulty. I sincerely appreciate the encouragement extended to me by the Head of the Department of Electronics & Telecommunication, Prof (Dr.) Hassanali G. Virani and our Principal, Dr. M. S. Krupashankara for allowing me to use the resources of the college that were immensely useful in carrying out the work related to my project. I also need to thank my parents, family members, and all well wishers along with the almighty without whom this work would not have taken shape.

REFERENCES

[1] Muhammad Adeel Pasha, Steven Derrien, Olivier Sentieys, "Toward Ultra Low-Power Hardware Specialization of a Wireless Sensor Network Node," 2009 IEEE 13th International Multitopic Conference.

[2] Cuong M. Nguyen, Jeffrey Mays, Dakota Plesa, Smitha Rao, Minh Nguyen, J.-C. Chiao, "Wireless Sensor Nodes for Environmental Monitoring in Internet of Things," IEEE MTT-S International Microwave Symposium, pp. 1 – 4, 2015

[3] Leonardo Barboni, Maurizio Valle, "Experimental Analysis of Wireless Sensor Nodes Current Consumption," Second International Conference on Sensor Technologies and Applications (sensorcomm 2008), pp. 401 – 406, 2008

[4] S. Rhee, D. Seetharam , S. Liu, "Techniques for minimizing power consumption in low data-rate wireless sensor networks," IEEE Wireless Communications and Networking Conference (IEEE Cat. No.04TH8733),vol.3, pp.1727 – 1731, 2004

[5] Alain Pegatoquet, Trong Nhan Le, Michele Magno, "A Wake-Up Radio-Based MAC Protocol for Autonomous Wireless Sensor Networks," IEEE/ACM Transactions on Networking, vol.27, issue 1, pp. 56 – 70, 2019

[6] Fariborz Entezami, Christos Politis, "Deploying parameters of Wireless Sensor Networks in test bed environment," IEEE Wireless Communications and Networking Conference Workshops (WCNCW), pp.145–149,2014