JSS ACADEMY OF TECHNICAL EDUCATION

Bangalore

PROJECT SYNOPSIS

Web Based Monitoring of Soil Parameters Using Constrained Application Protocol (CoAP)

Batch number: B2

Project group members:

Girish E. (1JS10EC031)
Pramod S. (1JS10EC066)
Prashant Aithal (1JS10EC067)
Rajath B.R. (1JS10EC071)

Under the guidance of:

Mrs. Kavitha H S
Asst. Professor
Dept. of Electronics and Communication Engineering

Signature of the Project Coordinator **Signature of the Guide**

Web Based Monitoring of Soil Parameters Using Constrained Application Protocol (CoAP)

Introduction

Wireless Sensor Networks have become the ideal candidate to provide effective and economically viable solutions for a large variety of applications ranging from health monitoring, scientific data collection, environmental monitoring to military operations. Wireless Sensor Networks are typically made of resource constrained devices that are low-cost, low-power, low bit-rate supporting short-range communications.

Literature Survey

In recent years, Wireless Sensor Networks has captivated the attention of researchers considering its potential application in wide variety of areas to observing the real world phenomena.

- There are a number of standardization bodies in the field of WSNs. The IEEE focuses on the physical and MAC layers; the Internet Engineering Task Force works on layers 3 and above. In addition to these, bodies such as the International Society of Automation provide vertical solutions, covering all protocol layers.
- The Internet of Things refers to uniquely identifiable objects and their virtual representations in an Internet-like structure. The term Internet of Things was proposed by Kevin Ashton in 1999, though the concept has been discussed in the literature since at least 1991.
- The Internet Engineering Task Force (IETF) Constrained RESTful environments Working Group has done the major standardization work for Constrained Application Protocol.

Objective

A proof-of-concept WSN is presented, to collect soil parameters, which is one of the most fundamental data required for precision agriculture. Monitoring of Wireless Sensor Networks using Constrained Application Protocol (CoAP) will be undertaken. In this context, measurement of the following soil parameters will be done,

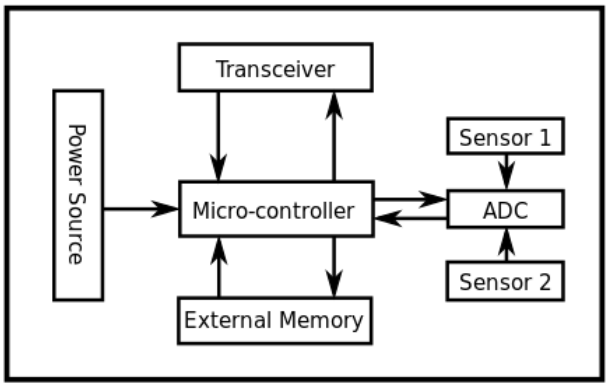
- Electrical Conductivity
- Volumetric Water Content
- Soil Temperature

Overview

Agriculture Sensor Network Setup

1) ***Hardware***-A **sensor node**, also known as **mote**, is a node in a wireless sensor network that is capable of performing some processing, gathering sensory information and communicating with other connected nodes in the network.

The **TelosB mote** is used which consists of Texas Instruments **MSP430 Microcontroller** along with 250kbits/s 2.4 GHz IEEE 802.15.4 Chipcon **Wireless Transceiver.** It has 10kB RAM along with 48kB flash.



Typical architecture of Sensor Node(Mote)

The **5TE soil sensor** from Decagon device is used as a sensor. This soil sensor is interfaced to work with TelosB for the application demonstration. 5TE is an integrated sensor that can measure 3 different soil parameters - **electrical conductivity** (EC), **soil temperature** and **volumetric water content.** The output from the sensor is in the form of raw data in Tera Term Language (TTL) format as given below:

56 432 645<0D>zG<0D><0A>

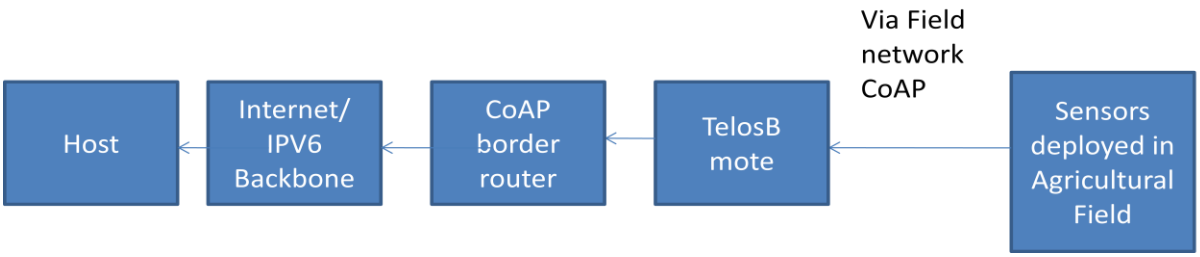
The raw data requires to be converted with suitable formula to arrive at the actual soil parameters.

2) ***Software***: Contiki 2.7 embedded OS release is used with TelosB motes in the project. Contiki is an open source, highly portable, multi-tasking operating system for the resource constrained wireless sensor networks. The Contiki OS kernel is event driven and it is completely written in C programming language, and supports dynamic runtime linking of application programs. In Contiki 2.7, platforms with the TI MSP430 can be emulated.

Internet Protocol version 6 (**IPv6**) is the latest revision of the Internet Protocol, the communications protocol that provides an identification and location system for computers on networks. Ipv6 has several advantages over IPv4.

Constrained Application Protocol (CoAP) is an application layer protocol that is intended for use in resource-constrained internet devices, such as WSN nodes. CoAP is designed to easily translate to HTTP for simplified integration with the web, while also meeting specialized requirements such as multicast support, very low overhead, and simplicity.

Contiki Erbium CoAP implementation that is bundled with Contiki 2.7 is used in the development. The Erbium REST engine includes framework for developing both CoAP server and CoAP client applications. The release also includes a **Mozilla Firefox browser plugin Copper**, CoAP user agent implementation for monitoring resources using Web browser.



Block diagram

The block diagram shows the monitoring of remote agricultural sensor resources using CoAP. The architecture has two network segments, namely ***Agricultural field network*** and ***Monitoring Network***. Field network consist of motes that are interfaced to soil sensors. The motes run CoAP server managing the monitorable CoAP resources. The soil sensor resources and onboard sensor resources can be discovered and CoAP methods can be acted on them using the identified CoAP URI. The CoAP application network will be connected to the IPv6 backbone/internet using the CoAP Proxy/border router. The CoAP proxy is the gateway for the field network to connect internet using cellular network.

Monitoring network consists of CoAP client node(s) and web server. CoAP clients can be utilized for real-time access to sensor data, while web server can give access archived data to the farmer and the agricultural scientist.

Future Scope

The number of devices that will connect to the Future Internet is increasing exponentially. The 6LoWPAN adaptation layer enables assignment of IPv6 addresses to low-power wireless devices making them reachable from any other node on the internet. Here we have shown how CoAP open application layer protocol can be utilized for real-time monitoring of IP-enabled agriculture sensors network.

Future scope includes the field deployment of the CoAP-based agriculture sensors network and its connectivity to internet IPv6 backbone for real-time monitoring over the internet.

Methodology:

• Hardware & Software Specifications
• Design
• Coding
• Testing
• Debugging

Tools used: Contiki OS, TelosB mote (TI MSP430F1611), Cooja Simulator and 5TE soil sensor.

References

[1] “Leveraging CoAP towards monitoring agriculture sensors network”-A. Paventhan, Sai Krishna Allu, V. Gayathri, Sameer Barve and N. Mohan Ram -ERNET India R&D Centre, (An autonomous scientific society under the Ministry of Communications & Information Technology, Government of India).

[2]” A Prototype Wireless Sensor Network for Precision Agriculture”-Jao, J. ; Bo Sun ; Kui Wu, Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops.

[3]” MSP430 Microcontroller Basics”-John H. Davies

[4]” Embedded Systems: A Contemporary Design Tool”, 1st Edition-James K. Peckol

[5] IEEE Std 802.15.4 Specifications for Low-Rate Wireless Personal Area Networks WPANs. IEEE Computer Society, September 2006.

[6]”Data Communications and Networking”- Behrouz A Forouzan, Fourth Edition- McGraw-Hill Higher Education.

ABSTRACT

Wireless sensor networks are widely deployed in many areas such as military, mining, health care, agriculture etc. These WSN consists of many small, low power, intelligent sensor nodes (motes) and one or more base stations. WSN networks generally operate in areas to which man does not have access. The distinguishing factor of WSNs is that they operate unattended over a period of time. These features have led to the success of the WSNs in real world deployment. WSNs are typically made of resource constrained devices that are low-cost, low-power and low-bitrate supporting short-range communications.

Towards realizing the broader vision of Internet of Things (IoT), various standards are emerging which provide end-to-end connectivity between resource-constrained devices. The "Internet of Things" describes a vision where objects become part of the Internet: where every object is uniquely identified, and accessible to the network, its position and status known, where services and intelligence are added to this expanded Internet, fusing the digital and physical world, ultimately impacting on our professional, personal and social environments.

In the project, the concepts of WSN have been applied to precision agriculture wherein soil parameters are measured in remote locations and make it available to the farmers. This allows the farmers to analyze the parameters and determine the condition of the soil and initiate steps if necessary.

Contiki OS, which is a lightweight operating system with support for dynamic loading and replacement of individual programs and services, has been used. The project aims to help the agriculturists to embrace technology to minimize and optimize their experience of farming.

TABLE OF CONTENTS

CHAPTER- 1

INTRODUCTION

1. Introduction

The world has shrunk in size is a saying heard everywhere. This has been possible because of communication. Communication is what defines our day-to-day activities. Communication technology encompasses a broad range of mediums, from the internet to radio to television to wireless signal providers. It is used in the business sphere, in personal relationships and also in public spaces that are neither primarily commercial nor personal, such as a subway stop that uses televisions to broadcast schedule changes. Traditionally, communication technology is limited to hardware such as radio receptors or television sets. However, the popularity of wireless technologies has correspondingly made the concept of communication technology slightly more ethereal. Wireless Sensor Networks is one such technology.

1.1. Need for Wireless Sensor Networks

All the things that were previously inaccessible have become accessible due to Wireless Sensor Networks. In recent times, wireless sensor networks have been widely deployed in many areas such as military, mining, health care, agriculture etc. WSN networks generally operate in areas where a man cannot reach for activities such as structural monitoring, environment monitoring etc., and most of WSNs operate unattended over a period.

WSNs operate in complex and noisy real-time, real-world applications. Whatever we thought was impossible before is possible because of WSNs. Home automation systems, monitoring of parameters pertaining to agriculture, geography, environment etc., at low cost has been possible because of WSNs. Therefore, we have explored WSN to apply to precision agriculture as well.

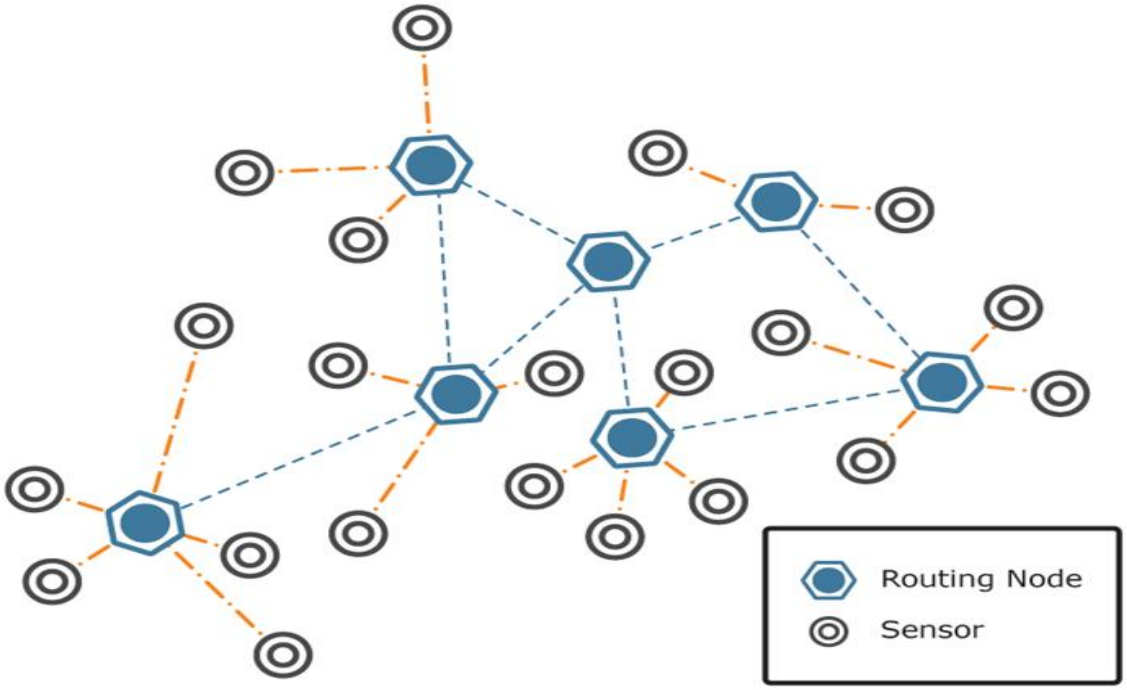


Figure 2: Sensor Nodes

1.2. Project Objective

The objective of our project is to implement a wireless sensor network using IPv6 connectivity to facilitate precision agriculture. Farmers can view the soil parameters from remote locations measured by the sensors via the internet. There is a need for real time collection of data for precision agriculture to be realized. By deploying motes which are integrated with sensors and integrating these motes to the internet, farmers will be able to view the parameters from their desktops/mobiles. The soil parameters measured are the soil temperature, electrical conductivity and volumetric water content of the soil. Based on these parameters, farmers can determine the health of the soil.

1.3. Scope

The potential of WSN monitoring can enable important new class of applications such as structural monitoring, office/home automation, automatic weather observation systems, patient health monitoring, agricultural field measurements and automated early warnings to an impending disaster. Sensor networks comprise of collection of nodes individually observing a phenomena in close range and collectively, they can observe properties of a larger area by employing appropriate communication topology.

1.4. Challenges

Wireless sensor networks also present a number of challenges a few of which are listed below-

- **Restricted resources.** Due to limited resources, the software components to be deployed in the motes should be lightweight. In addition, since it is anticipated that a WSN will execute multiple applications concurrently, it is very likely that performance requirements of all the running applications cannot be simultaneously satisfied. Therefore, it is necessary to provide mechanisms to optimize resource allocation and smartly trade the QoS of various applications against each other.
- **Network dynamics**. As an ad hoc network, a WSN may exhibit a highly dynamic topology due to mobility, communication failures or node failures. Programming paradigms and middleware should support the robust operation of WSN s despite these dynamics by adapting to the changing network environment.
- **Scale of deployments**. As stated before, a WSN is thought to consist of hundreds or thousands of nodes. In this sense, cluster-based architectures promote a more efficient use of resources in controlling large dynamic networks.
- **Data centric**. With the large population of sensor nodes, it may be impractical to pay attention to each individual node. Applications will focus on what data is desired rather than on individual sensor nodes. For example, users would be more interested in querying which area(s) has(have) a temperature higher than 30°C, or what the average temperature is in the southeast quadrant, rather than the temperature at sensor number 57.

- **Collection and processing of sensed data**. Most WSN applications involve nodes that contain redundant data and are located in a specific local region. This open up the possibility of in-network aggregation of data from different sources, eliminating redundancy and minimizing the number of transmissions to the sink. This saves considerable energy and resources, given that communication costs are much higher than computation costs.

CHAPTER- 2

LITERATURE SURVEY

2. Literature Survey

2.1. Internet of things

In a 2005 report the International Telecommunications Union (ITU) suggested that the "Internet of Things will connect the world's objects in both a sensory and intelligent manner". By combining various technological developments, the ITU has described four dimensions in IoT: *item identification* ("tagging things"), *sensors and wireless sensor networks* ("feeling things"), *embedded systems* ("thinking things") and *nanotechnology* ("shrinking things").

The definition of "things" in the IoT vision is very wide and includes a variety of physical elements. These include personal objects we carry around such as smart phones, tablets and digital cameras. It also includes elements in our environments (be it home, vehicle or work) as well as things fitted with tags (RFID or other) which become connected via a gateway device (e.g. a smart phone). Based on the above view of "things" an enormous number of devices and things will be connected to the Internet, each providing data and information and some, even services.

The IoT vision enhances connectivity from "any-time, any-place" for "any-*one"* into "any-time, any-place" for "any-*thing"*. Once these things are plugged into the network, more and more smart processes and services are possible which can support our economies, environment and health.

Figure 2 provides a view of the IoT ecosystem. Things could be tagged, and through scanners, identified, and the relevant location information could be communicated. Similarly, networked things with sensors become smaller, weaving themselves into our daily lives, while sensor and actuator networks act on the local environment, communicating status and events to a higher level service. Smart things sense activity and status, linking it to the IoT. Middleware and frameworks enabling application and service development which utilize data as received from (or about) things, most often living in the cloud provide the capability to add intelligence resulting in better services, which ultimately impact on the environment.

It is projected that almost everything will be connected to the network, even individual objects will be tracked, its condition and location communicated in real time to a higher level service.

In this context, it is important to note the envisioned scale of the IoT. Billions of things are network connected, each one providing data, many of them with the ability to act and influence their environment. Once these masses of data are intelligently processed, smarter services enhancing decision making and action could be created.

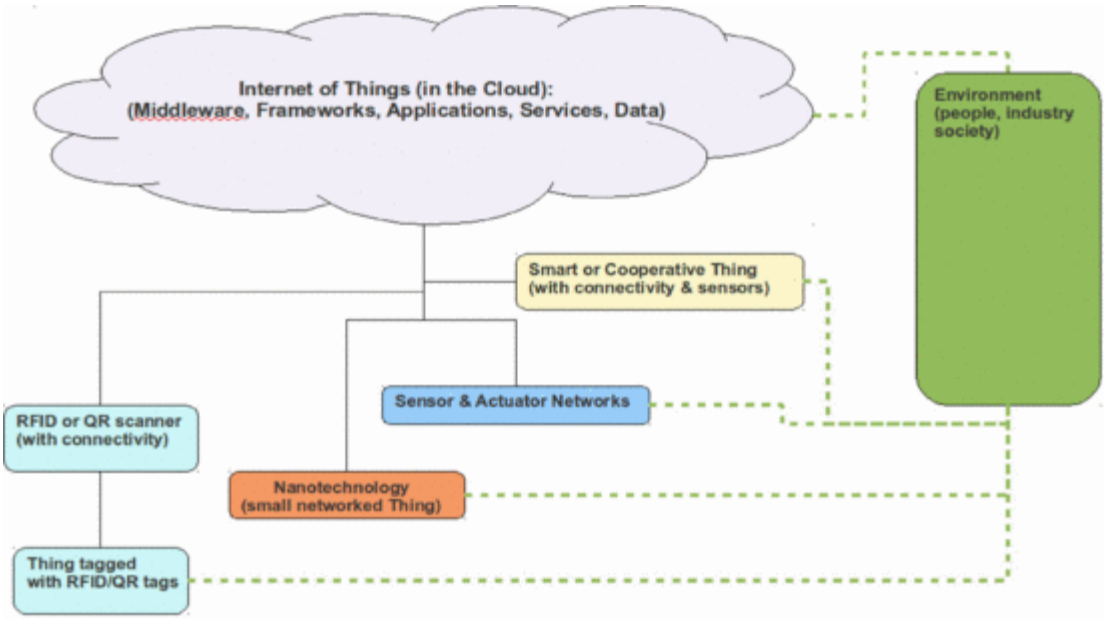


Figure 2: The Internet of Things Ecosystem

2.1 Precision Agriculture

The concept of precision agriculture has been around for some time now. Blackmore (1994) defined it as a "comprehensive system designed to optimize agricultural production by carefully tailoring soil and crop management to correspond to the unique condition found in each field while maintaining environmental quality". Nowadays, it is possible to say that precision agriculture (Zhang et al., 2002) is a method for estimating, assessing and understanding the changes that take place in crops in order to be able to determine irrigation and fertilizer requirements, product growth and ripening phases, optimum points of sowing and harvesting, etc. as exactly as possible—in other words, adequately predict the various stages in crop production. To that end it is important to gather as much information as possible on the water, soil, plants and environment. Many wireless technologies have been put to different uses to implement wireless sensors in precision agriculture from simple infra-red-based devices (IrDAs) for very short distances to long-range systems based on mobile telephony, such as GSM/GPRS. In between there are WPANs (Wireless Personal Area Networks) for short distances, such as Bluetooth (10m) and ZigBee (70m), and WLANs (Wireless Local Area Networks) for intermediate distances (100 m).

2.3.3 RPL

Low-power and Lossy Networks (LLNs) consist largely of constrained nodes (with limited processing power, memory, and sometimes energy when they are battery operated or energy scavenging). These routers are interconnected by lossy links, typically supporting only low data rates that are usually unstable with relatively low packet delivery rates. Another characteristic of such networks is that the traffic patterns are not simply point-to-point, but in many cases point-to- multipoint or multipoint-to-point. Furthermore, such networks may potentially comprise up to thousands of nodes. RPL was designed with an objective to meet such requirements.

In order to be useful in a wide range of LLN application domains, RPL separates packet processing and forwarding from the routing optimization objective. Examples of such objectives include minimizing energy, minimizing latency, or satisfying constraints.

RPL operations require bidirectional links. In some LLN scenarios, those links may exhibit asymmetric properties. It is required that the reachability of a router be verified before the router can be used as a parent. RPL expects an external mechanism to be triggered during the parent selection phase in order to verify link properties and neighbour reachability. Neighbour Unreachability Detection (NUD) is such a mechanism, but alternates are possible, including Bidirectional Forwarding Detection (BFD) [RFC5881] and hints from lower layers via Layer 2 (L2) triggers like [RFC5184]. In a general fashion, a detection mechanism that is reactive to traffic is favoured in order to minimize the cost of monitoring links that are not being used. RPL also expects an external mechanism to access and transport some control information, referred to as the "RPL Packet Information", i.e data packets.

RPL routes are optimized for traffic to or from one or more roots that act as sinks for the topology. As a result, RPL organizes a topology as a Directed Acyclic Graph (DAG) that is partitioned into one or more Destination Oriented DAGs (DODAGs), one DODAG per sink. If the DAG has multiple roots, then it is expected that the roots are federated by a common backbone, such as a transit link.

RPL uses four values to identify and maintain a topology:

- RPL Instance ID- A RPL Instance ID identifies a set of one or more

Destination Oriented DAGs (DODAGs). A network may have multiple RPL Instance IDs, each of which defines an independent set of DODAGs, which may be optimized for different Objective Functions (OFs) and/or applications. The set of DODAGs identified by a RPL Instance ID is called a RPL Instance. All DODAGs in the same RPL Instance use the same OF.

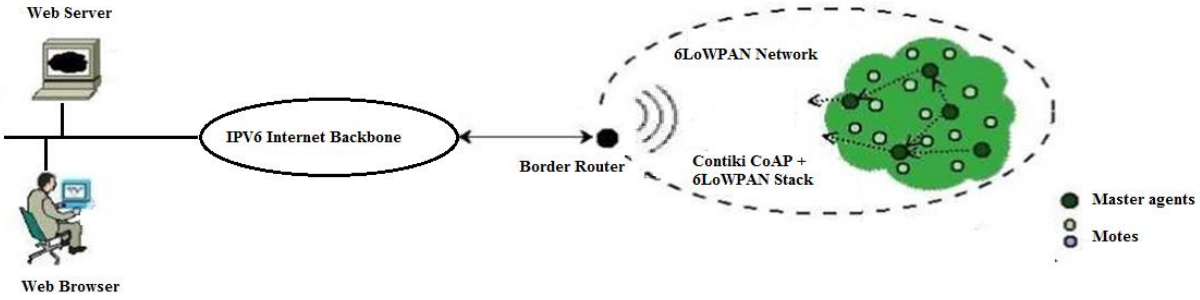
- The second is a DODAGID. The scope of a DODAGID is a RPL

Instance. The combination of RPL Instance ID and DODAGID uniquely identifies a single DODAG in the network. A RPL Instance may have multiple DODAGs, each of which has an unique DODAGID.

- The third is a DODAG Version Number- The scope of a

DODAG Version Number is a DODAG. A DODAG is sometimes reconstructed from the DODAG root, by incrementing the DODAG Version Number. The combination of RPL Instance ID, DODAGID, and DODAG Version Number uniquely identifies a DODAG Version.

- The fourth is Rank- The scope of Rank is a DODAG Version. Rank establishes a partial order over a DODAG Version, defining individual node positions with respect to the DODAG root.



The Proposed CoAP monitoring architecture for agricultural application is shown in Figure 4. The architecture shows the remote monitoring of agricultural sensor resources using CoAP. The architecture has two network segments, namely Agricultural field network and Monitoring Network. Field network consists of motes that are interfaced to soil sensors.

The motes run CoAP server managing the monitorable CoAP resources. The soil sensor resources can be discovered and CoAP methods can be acted on them using the identified CoAP URL. The CoAP application network will be connected to the ERNET IPv6 backbone/internet using the CoAP Proxy/border router. The CoAP proxy is the gateway for the field network to connect internet using cellular network. Monitoring network consists of CoAP client node(s) and web server. CoAP clients can be utilized for real-time access to sensor data, while web server can give access of the archived data to the farmer and the agricultural scientists as and when required.

CHAPTER-3

HARDWARE

3. HARDWARE

3.1. 5te Soil Sensors

The 5TE soil sensors will measure water content, electrical conductivity and temperature of the soil. The 5TE uses an oscillator running at 70 MHz to measure the dielectric permittivity of soil to determine the water content. A thermistor in thermal contact with the sensor prongs provides the soil temperature, while the screws on the surface of the sensor form a two-sensor electrical array to measure electrical conductivity. The Polyurethane coating on the 5TE circuit board protects the components from water damage and gives the sensor a longer life span.

Figure 4: 5TE Soil Sensor

THEORY:

Volumetric Water Content

The 5TE sensor uses an electromagnetic field to measure the dielectric permittivity of the surrounding medium. The sensor supplies a 70 MHz oscillating wave to the sensor prongs that charges according to the dielectric of the material. The stored charge is proportional to soil dielectric and soil volumetric water content. The 5TE microprocessor measures the charge and outputs a value of dielectric permittivity from the sensor.

Temperature

The 5TE uses a surface-mounted thermistor to take temperature readings. The thermistor is underneath the sensor over mold, next to one of the prongs, and it reads the temperature of the prong surface. The 5TE outputs temperature in ˚C.

Electrical Conductivity

Electrical conductivity (EC) is the ability of a substance to conduct electricity and can be used to infer the amount of polar molecules that are in solution. EC is measured by applying an alternating electrical current to two electrodes and measuring the resistance between them. Conductivity is then derived by multiplying the inverse of the resistance (conductance) by the cell constant (the ratio of the distance between the electrodes to their area).

The 5TE uses a two-sensor array to measure the EC. The array is located on the screws of two of the 5TE prongs.

The 5TE uses a two electrode array to measure the bulk EC of the surrounding medium. Decagon factory calibrates the bulk EC measurement to be accurate within ±10% from 0 to 7 dS/m. This range is adequate for most field, greenhouse and nursery applications. However, some special applications in salt affected soils may require measurements with bulk EC greater than the specified range. The 5TE can measure up to 23.1 dS/m bulk EC, but requires user

calibration above 7 dS/m. additionally; EC measurements above 7 dS/m are sensitive to contamination of the electrodes by skin oils, etc.

Converting Bulk EC to Pore EC

For many applications, it is advantageous to know the electrical conductivity of the solution contained in the soil pores (σ_p), which is a good indicator of the solute concentration in the soil. Researchers have traditionally obtained σ_p by extracting pore water from the soil and measuring σ_p directly. However, this is a time consuming and labor intensive process.

The 5TE measures the electrical conductivity of the bulk soil surrounding the sensors (σ_b). We have conducted a considerable amount of research to determine the relationship between σ_b and σ_p. Recent work by Hilhorst (2000) takes advantage of the linear relationship between the soil bulk dielectric permittivity (ε_b) and σ_b to allow accurate conversion from σ_b to σ_p if you know the ε_b. The 5TE measures ε_b and σ_b nearly simultaneously in the same soil volume, so it is well suited to this method.

THEORY

Use Hilhorst, 2000 to derive the pore water conductivity.

$$\sigma_p = \frac{\varepsilon_p \sigma_b}{\varepsilon_b - \varepsilon_{\sigma b=0}}$$

where σ_p is the pore water electrical conductivity (dS/m); ε_p is the real portion of the dielectric permittivity of the soil pore water (unitless); σ_b is the bulk electrical conductivity, (dS/m), measured directly by the 5TE; ε_b is the real portion of the dielectric permittivity of the bulk soil (unitless); $\varepsilon_{\sigma b=0}$ is the real portion of the dielectric permittivity of the soil when bulk electrical conductivity is 0 (unitless).

ε_p can be calculated from soil temperature using a simple formula.

$\varepsilon_p = 80.3 - 0.37 * (T_{soil} - 20)$

The 5TE measures T_{soil} or soil temperature (°C) and ε_b. You can convert raw VWC counts to bulk dielectric with the 5TE dielectric calibration.

$$\varepsilon_b = \frac{\varepsilon_{Raw}}{50}$$

Finally, $\varepsilon_{\sigma b=0}$ is an offset term loosely representing the dielectric of the dry soil. Hilhorst (2000) recommends using $\varepsilon_{\sigma b=0} = 4.1$ as a generic offset. However, our research in several agricultural soils, organic, and inorganic growth media indicates that $\varepsilon_{\sigma b=0} = 6$ results in more accurate determinations of σ_p. Hilhorst (2000) offers a simple and easy method for determining for individual soil types, which will improve the accuracy of the calculation of σ_p in most cases.

Our testing indicates that the above method for calculating σ_p results in good accuracy (±20%) in moist soils and other growth media. In dry soils where VWC is less than about 0.10 m^3/m^3, the denominator of pore water conductivity equation becomes very small, leading to large potential errors. We recommend you not use this method to calculate σ_p in soils with VWC < 0.10 m^3/m^3.

Pore Water Versus Solution EC

As noted in the section on "Converting Bulk EC to Pore EC," we can calculate pore water electrical conductivity from bulk EC.

THEORY

Pore water EC is the electrical conductivity of the water in the pore space of the soil. One could measure this directly by squeezing the soil under high pressure to force water out of the soil matrix and test the collected water for EC.

Solution EC is the electrical conductivity of pore water removed from a saturated paste. In this case, wet the soil with distilled water until the soil saturates, then place the soil on filter paper in a vacuum funnel and apply suction. An electrical conductivity measurement on the removed sample water gives the solution electrical conductivity. Theoretically, the two are related by the

bulk density. An example calculation illustrates this relationship. If a soil is at 0.1 m^3/m^3 VWC, has pure water EC of 0.7 dS/m, and a bulk density of 1.5 Mg/m^3. We can calculate the solution EC with two equations.

$$\phi = 1 - \frac{\rho_b}{\rho s} = 1 - \frac{1.5}{2.65} = 0.43$$

$$\text{Solution EC} = \frac{\sigma_p \theta + \sigma_d(\phi - \theta)}{\phi} = \frac{0.7(0.1) + 0}{0.43} = 0.162 \text{ dS/m}$$

In this example, φ is the porosity, ρ_b is bulk density, ρ_s is the density of the minerals (assumed to be 2.65 Mg/m^3), the subscript d is distilled water, and θ is volumetric water content. We assume that the EC of the distilled water is 0 dS/m. In practice, solution EC calculated from this method and solution EC taken from a laboratory soil test may not correlate because wetting soil to a saturated paste is very imprecise.

3.1.2. Calibration

Dielectric Permittivity

Decagon factory calibrates each 5TE sensor to measure dielectric permittivity (ε_a) accurately in the range of 1 (air) to 80 (water). The unprocessed raw values reported by the 5TE in standard serial communication have units of $\varepsilon_a * 50$. When used in SDI- 12 communication mode, the unprocessed values have units of ε_a (for 5TE board versions R2-04 and older, units are, $\varepsilon_a * 100$).

Mineral Soil Calibration

Numerous researchers have studied the relationship between dielectric permittivity and volumetric water content (VWC) in soil. As a result, numerous transfer equations that predict VWC from measured dielectric permittivity. You are free to use any of these various transfer equations to convert raw dielectric permittivity data from the 5TE into VWC.

$$VWC = 4.3 * 10^{-6}\varepsilon^3_a - 5.5 * 10^{-4}\varepsilon^2_a + 2.92 * 10^{-2}\varepsilon_a - 5.3 * 10^{-2}$$

In a properly installed 5TE sensor in a normal mineral soil with saturation extract electrical conductivity < 10 dS/m, the Topp equation results in measurements within ±3% VWC of the actual soil VWC. If you require more accurate VWC than ±3%, are working in a soil with very high electrical conductivity, or non-normal mineralogy, then it may be necessary to conduct a soil specific calibration of the 5TE sensor to improve the accuracy to 1 to 2% for any soil.

Measurement Specifications:

VWC:

Accuracy: **ε_a:** ±1 ε_a (unitless) from 1-40 (soil range), ±15% from 40-80

- Using Topp equation: ±0.03 m^3/m^3 (±3% VWC) typical in soils that have solution electrical conductivity < 10 dS/m
- Using medium specific calibration: ±0.01 - 0.02 m^3/m^3 (± 1-2% VWC) in any porous medium.

Resolution: ε_a: 0.1 ε_a (unitless) from 1-20, < 0.75 ε_a (unitless) from 20-80

VWC: 0.0008 m^3/m^3 (0.08% VWC) from 0 to 50% VWC

Range: Apparent dielectric permittivity (ε_a): 1 (air) to 80 (water)

Bulk Electrical Conductivity

Accuracy: ±10% from 0-7 dS/m, user calibration required above 7 dS/m

Resolution: 0.01 dS/m from 0-7 dS/m, 0.05 dS/m from 7-23 dS/m*

Range: 0-23 dS/m (bulk)

Temperature:

Accuracy: ±1°C

Resolution: 0.1°C

Range: -40°C to +50°C

3.1.3 GENERAL SPECIFICATIONS

Dimensions: 10 x 3.2 x 0.7 cm

Probe Length: 5.2 cm

Dielectric Measurement Frequency: 70MHZ

Measurement Time: 150 ms

Power: 3.6 - 15 VDC, 0.3 mA quiescent, 10 mA
during 150 ms measurement

Output: Serial TTL, 3 Volt Levels or SDI 12

Operating Temperature: -40°C to +50°C

Input / Output Circuitry

A single line is used for both transmit and receive. A 10 uH inductor and a 510 ohm resistor are in series with the RXD/TXD line on the microprocessor with a 220 pF capacitor to ground. The resistor and capacitor provide input protection. The inductor minimizes RF interference.

3.1.4. SENSOR EXCITATION

The sensor connects through a 3 wire cable with a stereo connector or bare wire interface.

The three connections are Excitation, Ground, and Serial Out (Data). The connector tip (White wire) is Excitation, the connector base (shield) is ground, and the intermediate ring (red wire) is Serial Out. The excitation is 3 to 15 volts. Current drain during the water content measurement (approximately 10 ms in duration) can be as high as 10 mA. After data transmission, the sensor will pull around 3mA for about 6 seconds. It will then go into a sleep state and draw around 30uA, until a SDI-12 break command is received. Be sure that the selected power supply can provide this current without being pulled below the 3.6 V level. If all sensors are excited at the same time, all of them will draw 10mA for 10ms. The system will need to source enough current to supply the

number of sensors in the system (up to 62), x, times 10mA (Total current = x * 10mA). See Figure 2 for an equivalent circuit diagram.

When excitation voltage is applied the sensor begins its measurement sequence. Within about 50 ms of excitation three values are transmitted on the serial out line.

3.1.5. TTL COMMUNICATION

The serial out uses 1200 baud asynchronous CMOS with 8 data bits, no parity, and one stop bit. The voltage levels are 0 - 3.6 V and the logic levels are TTL (active low). Each data byte consists of a start bit (low), 8 data bits (least significant bit first), and a stop bit (high).

3.1.6. TTL FORMAT DESCRIPTION

The data string output by the sensor should be in a format similar to the one below:

56 432 645<0D>zG<0D><0A>

Section	Description
56	Raw dielectric output in the format raw output = dielectric * 50. Values range from approximately 50 to 4094. To convert to VWC in mineral soil, we recommend the well-known Topp equation (Topp et al, 1980): $\theta = 4.3X10^{-6} * \varepsilon^3 - 5.5X10^{-4} * \varepsilon^2 + 2.92X10^{-2} * \varepsilon - 5.3X10^{-2}$ In this example, 56 this is the raw apparent dielectric reported. Dividing this by 50 gives a value of 1.12. This is an appropriate value for a sensor measuring air.
432	Electrical conductivity in mS/cm multiplied by 100. Divide this number by 100 to get mS/cm (or dS/m). This value is already temperature
	corrected within the 5TE probe using the temperature correction outlined by the US Salinity labs Handbook 60. Raw values for EC in tap water can range from 10 to 80 (0.1 to 0.8 dS/m). On 5TM's this value is zero and should be ignored. In this example, 432 is the raw bulk electrical conductivity reported. Dividing by 100 gives a value of 4.32 mS/cm. *Note: For raw values that exceed 700, the EC value needs to be decompressed first using the following equation:* $EC_{decompressed} = 5*(EC_{raw} - 700) + 700$

645	Temperature. This number is 10*T + 400, where T is the degrees Celsius. To convert it to temperature, subtract 400 and divide by 10. Room temperature gives a value between 600 and 650. In this example, 645 is the raw temperature value reported. Subtracting by 400 and dividing by 10 gives us a temperature of 24.5°C. *Note: For raw values that exceed 900, the T_{raw} needs to be decompressed prior to converting it to Celcius using the following equation:* $T_{decompressed} = 5*(T_{raw} - 900) + 900$
<0D>	This carriage return character signals the end of the measurement string and start of the meta data string.

Meta data:

z	Sensor Type. This character is used to indicate the sensor type. z is used for 5TE sensors, and x is used for 5TM sensors.
G	Checksum. This one character checksum is used in our instruments to ensure that the data transmitted are valid. The checksum is used for sections listed above: 56 432 645<0D>z See the following function for an example of how to implement the checksum algorithm in C.
<0D><0A>	The carriage return and line feed are used to signal the end of the meta data section and the end of the transmission.

Here is an example of how to calculate the checksum (crc) in C. In this case, the string passed to the function would be: "56 432 645<0D>z" and the returning value would be the character 'G'.

```
char CalculateChecksum(char * Response){
int length, sum = 0, i, crc;

// Finding the length of the response string
length = strlen(Response);

// Adding characters in the response together
for( i = 0; i < length; i++ )          sum += Response[i];

// Converting checksum to a printable character       crc = sum % 64 + 32;

return crc;
}
```

3.1.7. TTL TYPICAL OUTPUTS

Typical output while the sensor is suspended in air 20°C will be *approximately*:
50 0 600, which translates into a dielectric = 1, EC = 0, T = 20°C.
If the sensor is fully suspended in 20C tap water, the outputs will be *approximately*: 4000 50 600, which translates into dielectric = 80, EC = 0.5 dS/m, T = 20°C. Note that the raw dielectric output in water can range from 3400 to 4094, and that the EC of tap water is highly variable depending on the source.

3.2. SENSOR NODE

A sensor node, also known as a mote is a node in a wireless sensor network that is capable of performing some processing, gathering sensory information and communicating with other connected nodes in the network.

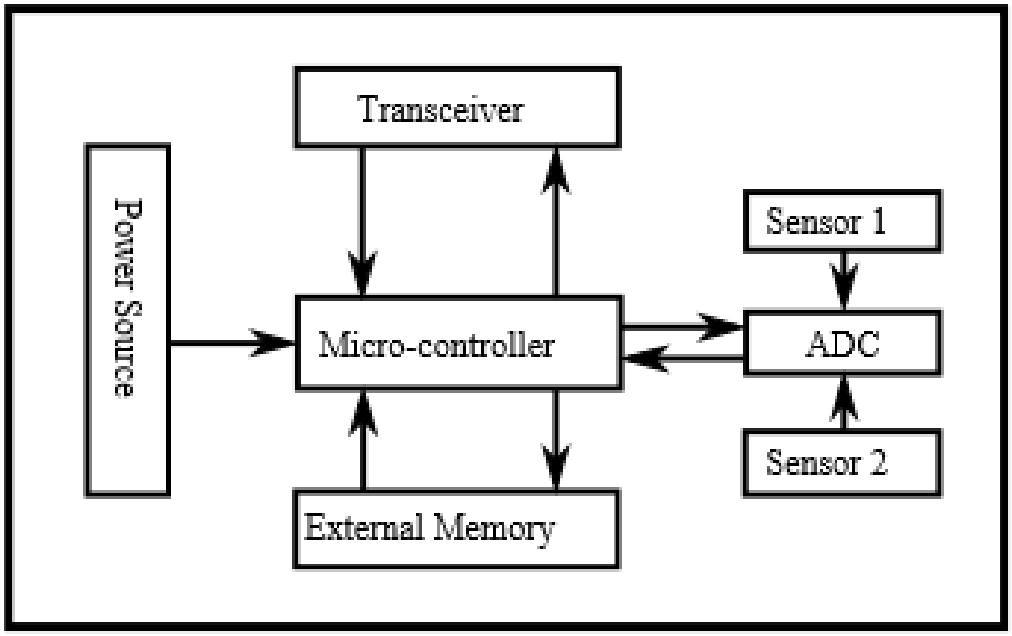


Figure 3: Block Diagram of Sensor Node

3.2.1. HISTORY

Although wireless sensor nodes have existed for decades and used for applications as diverse as earthquake measurements to warfare, the modern development of small sensor nodes dates back to the 1998 Smartdust project and the NASA Sensor Webs Project. One of the objectives of the Smartdust project was to create autonomous sensing and communication within a cubic millimetre of space. Though this project ended early on, it led to many more research projects.

They include major research centres in Berkeley NEST and CENS. The researchers involved in these projects coined the term *mote* to refer to a sensor node. The equivalent term in the NASA Sensor Webs Project for a physical sensor node is *pod*, although the sensor node in a Sensor Web can be another Sensor Web itself. Physical sensor nodes have been able to increase their capability in conjunction with Moore's Law. The chip footprint contains more complex and lower powered microcontrollers. Thus, for the same node footprint, more silicon capability can be packed into it. Nowadays, motes focus on providing the longest wireless range, the lowest energy consumption and the easiest development process for the user.

COMPONENTS OF A SENSOR NODE

The main components of a sensor node are a microcontroller, transceiver, external memory, power source and one or more sensors.

CONTROLLER

The controller performs tasks, processes data and controls the functionality of other components in the sensor node. While the most common controller is a microcontroller, other alternatives that can be used as a controller are: a general purpose desktop microprocessor, digital signal processors, FPGAs and ASICs. A microcontroller is often used in many embedded systems such as sensor nodes because of its low cost, flexibility to connect to other devices, ease of programming, and low power consumption.

A general purpose microprocessor generally has higher power consumption than a microcontroller; therefore it is often not considered a suitable choice for a sensor node. Digital Signal Processors may be chosen for broadband wireless communication applications, but in Wireless Sensor Networks the wireless communication is often modest: i.e., simpler, easier to process modulation and the signal processing tasks of actual sensing of data is less complicated. Therefore the advantages of DSPs are not usually of much importance to wireless sensor nodes. FPGAs can be reprogrammed and reconfigured according to requirements, but this takes more time and energy than desired.

3.2.2. TRANSCEIVER

Sensor nodes often make use of ISM band, which gives free radio, spectrum allocation and global availability. The possible choices of wireless transmission media are radio frequency (RF), optical communication (laser) and infrared. Lasers require less energy, but need line-of- sight for communication and are sensitive to atmospheric conditions. Infrared, like lasers, needs no antenna but it is limited in its broadcasting capacity. Radio frequency-based communication is the most relevant that fits most of the WSN applications. WSNs tend to use license-free communication frequencies: 173, 433, 868, and 915MHz; and 2.4 GHz.

The functionality of both transmitter and receiver are combined into a single device known as a transceiver. Transceivers often lack unique identifiers. The operational states are transmit, receive, idle, and sleep. Current generation transceivers have built-in state machines that perform some operations automatically.

Most transceivers operating in idle mode have a power consumption almost equal to the power consumed in receive mode. Thus, it is better to completely shut down the transceiver rather than leave it in the idle mode when it is not transmitting or receiving. A significant amount of power is consumed when switching from sleep mode to transmit mode in order to transmit a packet.

EXTERNAL MEMORY

From an energy perspective, the most relevant kinds of memory are the on-chip memory of a microcontroller and Flash memory—off-chip RAM is rarely, if ever, used. Flash memories are used due to their cost and storage capacity. Memory requirements are very much application dependent. Two categories of memory based on the purpose of storage are: user memory used for storing application related or personal data, and program memory used for programming the device. Program memory also contains identification data of the device if present.

3.2.3. POWER SOURCE

A wireless sensor node is a popular solution when it is difficult or impossible to run a mains supply to the sensor node. However, since the wireless sensor node is often placed in a hard-to-reach location, changing the battery regularly can be costly and inconvenient. An important aspect in the development of a wireless sensor node is ensuring that there is always adequate energy available to power the system. The sensor node consumes power for sensing, communicating and data processing. More energy is required for data communication than any other process. The energy cost of transmitting 1 Kb a distance of 100 metres (330 ft.) is approximately the same as that used for the execution of 3 million instructions by a 100 million instructions per second/W processor.

Power is stored either in batteries or capacitors. Batteries, both rechargeable and non-rechargeable, are the main source of power supply for sensor nodes. They are also classified according to electrochemical material used for the electrodes such as NiCd (nickel-cadmium), Ni-Zn (nickel-zinc), NiMH (nickel-metal hydride), and lithium-ion. Current sensors are able to renew their energy from solar sources, temperature differences, or vibration. Two power saving policies used are Dynamic Power Management (DPM) and Dynamic Voltage Scaling (DVS).DPM conserves power by shutting down parts of the sensor node which are not currently used or active. A DVS scheme varies the power levels within the sensor node depending on the non-deterministic workload. By varying the voltage along with the frequency, it is possible to obtain quadratic reduction in power consumption.

3.2.4. SENSORS

Sensors are hardware devices that produce a measurable response to a change in a physical condition like temperature or pressure. Sensors measure physical data of the parameter to be monitored. The continual analog signal produced by the sensors is digitized by an analog-to-digital converter and sent to controllers for further processing. A sensor node should be small in size, consume extremely low energy, operate in high volumetric densities, be autonomous and operate unattended, and be adaptive to the environment. As wireless sensor nodes are typically

very small electronic devices, they can only be equipped with a limited power source of less than 0.5-2 ampere-hour and 1.2-3.7 volts.

Sensors are classified into three categories: passive, omnidirectional sensors; passive, narrow-beam sensors; and active sensors. Passive sensors sense the data without actually manipulating the environment by active probing. They are self-powered; that is, energy is needed only to amplify their analog signal. Active sensors actively probe the environment, for example, a sonar or radar sensor, and they require continuous energy from a power source. Narrow-beam sensors have a well-defined notion of direction of measurement, similar to a camera. Omni-directional sensors have no notion of direction involved in their measurements.

3.2.5. SOFTWARE

Energy is the scarcest resource of WSN nodes, and it determines the lifetime of WSNs. WSNs are meant to be deployed in large numbers in various environments, including remote and hostile regions, where ad hoc communications are a key component. For this reason, algorithms and protocols need to address the following issues:

- Lifetime maximization
- Robustness and fault tolerance
- Self-configuration

Some of the important topics in WSN (Wireless Sensor Networks) software research are:

- Operating systems
- Security
- Mobility

3.2.6. OPERATING SYSTEMS

Operating systems for wireless sensor network nodes are typically less complex than general-purpose operating systems. They more strongly resemble embedded systems, for two reasons. First, wireless sensor networks are typically deployed with a particular application in mind, rather than as a general platform. Second, a need for low costs and low power leads most wireless sensor nodes to have low-power microcontrollers ensuring that mechanisms such as virtual memory are either unnecessary or too expensive to implement.

TinyOS is perhaps the first operating system specifically designed for wireless sensor networks. TinyOS is based on an event-driven programming model instead of multithreading. TinyOS programs are composed of event handlers and tasks with run-to-completion semantics. When an external event occurs, such as an incoming data packet or a sensor reading, TinyOS signals the appropriate event handler to handle the event. Event handlers can post tasks that are scheduled by the TinyOS kernel some time later.

Contiki is an OS which uses a simpler programming style in C while providing advances such as 6LoWPAN and Protothreads.

3.3. SIMULATION OF WSNS

At present, agent-based modelling and simulation is the only paradigm which allows the simulation of complex behaviour in the environments of wireless sensors. Agent-based simulation of wireless sensor and ad hoc networks is a relatively new paradigm. Agent-based modelling was originally based on social simulation.

Simulators like OPNET, NetSim, NS2 and Cooja can be used to simulate a wireless sensor network.

3.4. TELOSB MOTE

Telos is an ultra-low power wireless module for use in sensor networks, monitoring applications, and rapid application prototyping. Telos leverages industry standards like USB and IEEE 802.15.4 to interoperate seamlessly with other devices. Telos leverages emerging wireless protocols and the open source software movement. Telos is part of a line of modules featuring on-board sensors to increase robustness while decreasing cost and package size.

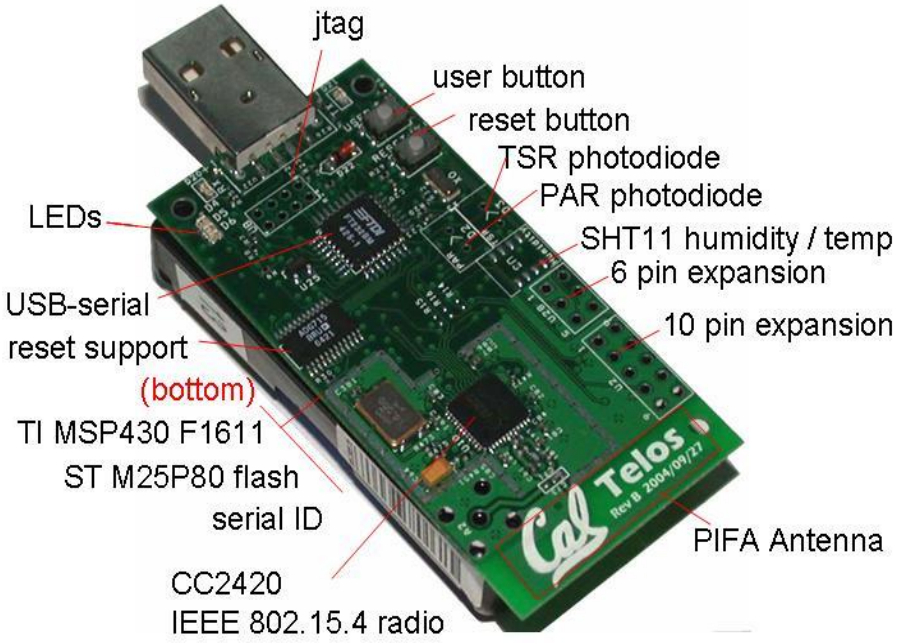


Figure 3: TelosB mote

Key Features

- 250kbps 2.4GHz IEEE 802.15.4 Chipcon Wireless Transceiver
- Interoperability with other IEEE 802.15.4 devices
- 8MHz Texas Instruments MSP430 microcontroller (10kB RAM, 48kB Flash)
- Integrated ADC, DAC, Supply Voltage Supervisor, and DMA Controller
- Integrated on board antenna with 50m range indoors / 125m range outdoors
- Integrated Humidity, Temperature, and Light sensors
- Ultra-low current consumption
- Fast wakeup from sleep (<6μs)

- Hardware link-layer encryption and authentication
- Programming and data collection via USB
- 16-pin expansion support and optional SMA antenna connector

Module Description

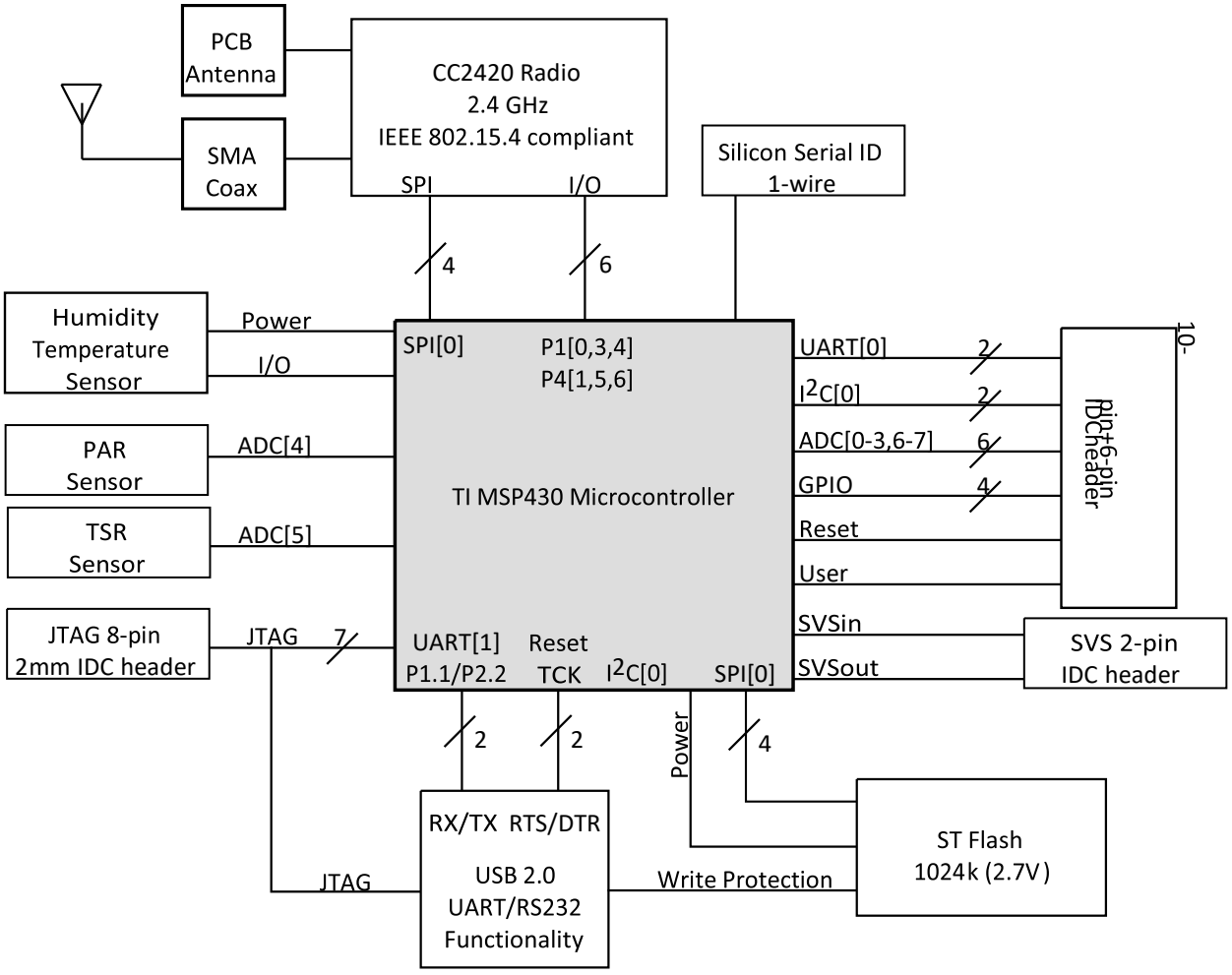
The Telos module is a low power "mote" with integrated sensors, radio, antenna, Microcontroller and programming capabilities.

Power

Telos may be powered by two AA batteries. The module was designed to fit the two AA battery form factor. AA cells may be used in the operating range of 2.1 to 3.6V DC, however the voltage must be at least 2.7V when programming the microcontroller flash or external flash.

If the Telos module is plugged into the USB port for programming or communication, it will receive power from the host computer. The mote operating voltage when attached to USB is 3V. If Telos will always be attached to a USB port, no battery pack is necessary. The 16-pin expansion connector can provide power to the module. Any of the battery terminal connections may also provide power to the module.

3.4.1. BLOCK DIAGRAM



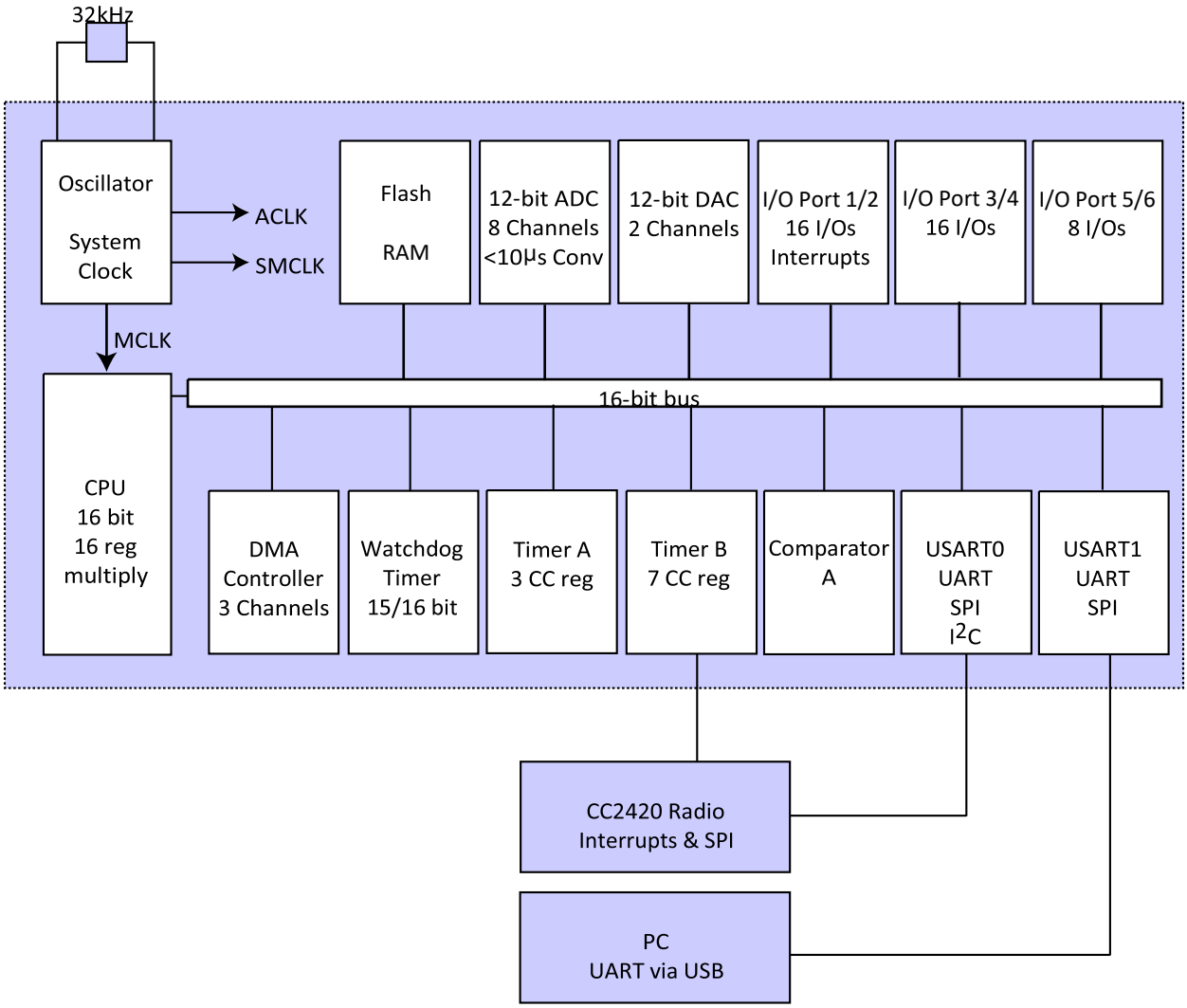
3.4.2. MICROPROCESSOR DESCRIPTION (MSP430)

The **MSP430** is a mixed-signal microcontroller family from Texas Instruments. Built around a 16-bit CPU, the MSP430 is designed for low cost and, specifically, low power consumption embedded applications.

The low power operation of the Telos module is due to the ultra-low power Texas Instruments MSP430 F1611 microcontroller featuring 10kB of RAM, 48kB of flash, and 128B of information storage. This 16-bit RISC processor features extremely low active and sleep current consumption that permits Telos to run for years on a single pair of AA batteries. The MSP430 has an internal digitally controlled oscillator (DCO) that may operate up to 8MHz. The DCO may be turned on from sleep mode in 6μs, however 292ns is typical at room temperature.

When the DCO is off, the MSP430 operates off an eternal 32768Hz watch crystal. Although the DCO frequency changes with voltage and temperature, it may be calibrated by using the 32kHz oscillator. In addition to the DCO, the MSP430 has 8 external ADC ports and 8 internal ADC ports. The ADC internal ports may be used to read the internal thermistor or monitor the battery voltage. A variety of peripherals are available including SPI, UART, digital I/O ports, Watchdog timer, and Timers with capture and compare functionality. The F1611 also includes a 2-port 12-bit DAC module, Supply Voltage Supervisor, and 3-port DMA controller.

3.4.3. BLOCK DIAGRAM OF MSP430



3.4.4. RADIO DESCRIPTION

Telos features the Chipcon CC2420 radio for wireless communications. The CC2420 is an IEEE 802.15.4 compliant radio providing the PHY and some MAC functions. With sensitivity exceeding the IEEE 802.15.4 specification and low power operation, the CC2420 provides reliable wireless communication. The CC2420 is highly configurable for many applications with the default radio settings providing IEEE 802.15.4 compliance. The CC2420 is controlled by the TI MSP430 microcontroller through the SPI port and a series of digital I/O lines and interrupts.

The radio may be shut off by the microcontroller for low power duty cycled operation.

External Flash

Telos Revision B uses the ST M25P80 40MHz serial code flash for external data and code storage. The flash holds 1024kB of data and is decomposed into 16 segments, each 64kB in size. The flash shares SPI communication lines with the CC2420 transceiver. Care must be taken when reading or writing to flash such that it is interleaved with radio communication, typically implemented as a software arbitration protocol for the SPI bus on the microcontroller.

3.4.5. EXPANSION CONNECTOR

Telos has two expansion connectors and a pair of onboard jumpers that may configured so that additional devices (analog sensors, LCD displays, and digital peripherals) may be controlled by the Telos module. On the far side of the board from the USB connector is a 10-pin IDC header at position U2 and a 6-pin IDC header at U28. The 10-pin connector has the same connections as Telos Revision A and is the primary connector. It provides digital input and output signals as well as and analog inputs. Peripherals may be connected to the 10-pin connector using an IDC header, an IDC ribbon cable, or by designing a printed circuit board that solders directly on to the IDC header providing a robust connection to the module. An additional 6-pin (U28) header provides access to the exclusive features of Revision B. Two additional ADC inputs are provided that may be reconfigured by software to be two 12-bit DAC outputs. ADC7 may also act as the input to the supply voltage supervisor. The user interface elements—the reset and user buttons—are exported by the 6-pin header for use in external interfaces and packaging.

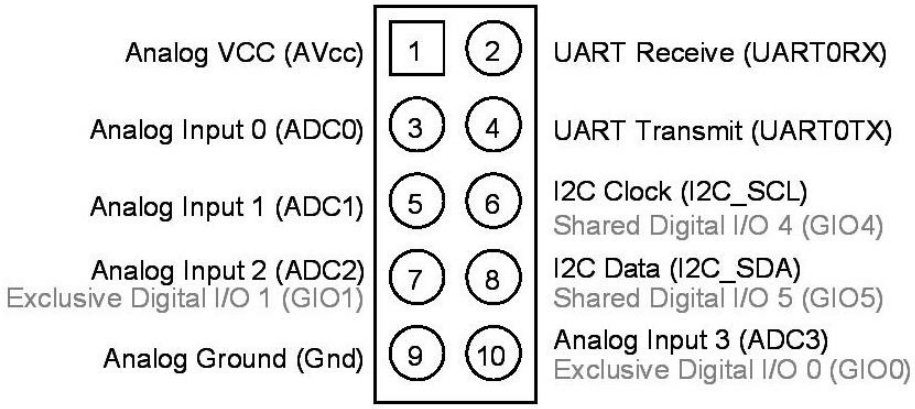


Figure 22 : Functionality of the 10-pin expansion connector (U2). Alternative pin uses are shown in gray.

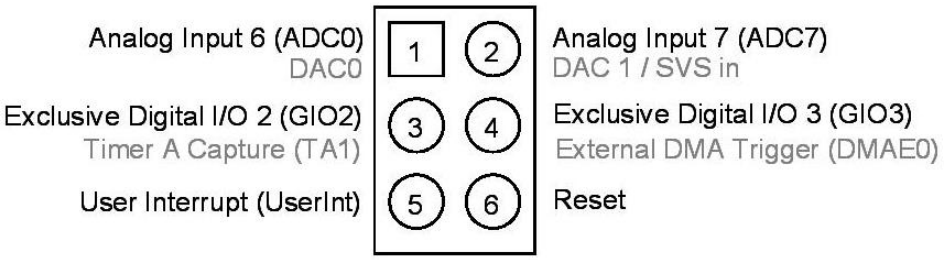


Figure 5: EXPANSION CONNECTOR

If expansion pin 10 (ADC3) is used for digital I/O instead of analog inputs, R14 must be populated with a 0 ohm resistor to enable the pin for digital I/O (GIO0) on the microcontroller. R16 must be populated with a 0 ohm resistor to enable GIO1. R14 and R16 are located on the top side of Telos between the USB controller and the radio.

The 6-pin IDC header also has an optional jumper, R15. By installing a 0 ohm resistor at R15, GIO3 is directly connected to SVSout. By making GIO3 an input and using the SVS features of the microcontroller, the SVSout function can be exported via pin 4 of U28.

A separate Supply Voltage Supervisor (SVS) 2-pin IDC header is provided underneath the USB connector at position U7. The SVS header allows add-on boards to be built that connect to the positive and negative battery terminals and the SVS pins in order to provide power the module and use the microcontroller's advanced SVS functionality for boost converters, solar systems, and rechargeable systems. The SVS header is shown in Figure 24 and includes the SVSin and SVSout pins from the microcontroller.

MSP430F1611 PIN DESIGNATION

The MSP430F15x/16x/161x series are microcontroller configurations with two built-in 16-bit timers, a fast 12-bit A/D converter, dual 12-bit D/A converter, one or two universal serial synchronous/asynchronous communication interfaces (USART), I2C, DMA, and 48 I/O pins. In addition, the MSP430F161x series offers extended RAM addressing for memory-intensive applications and large C-stack requirements.[3]

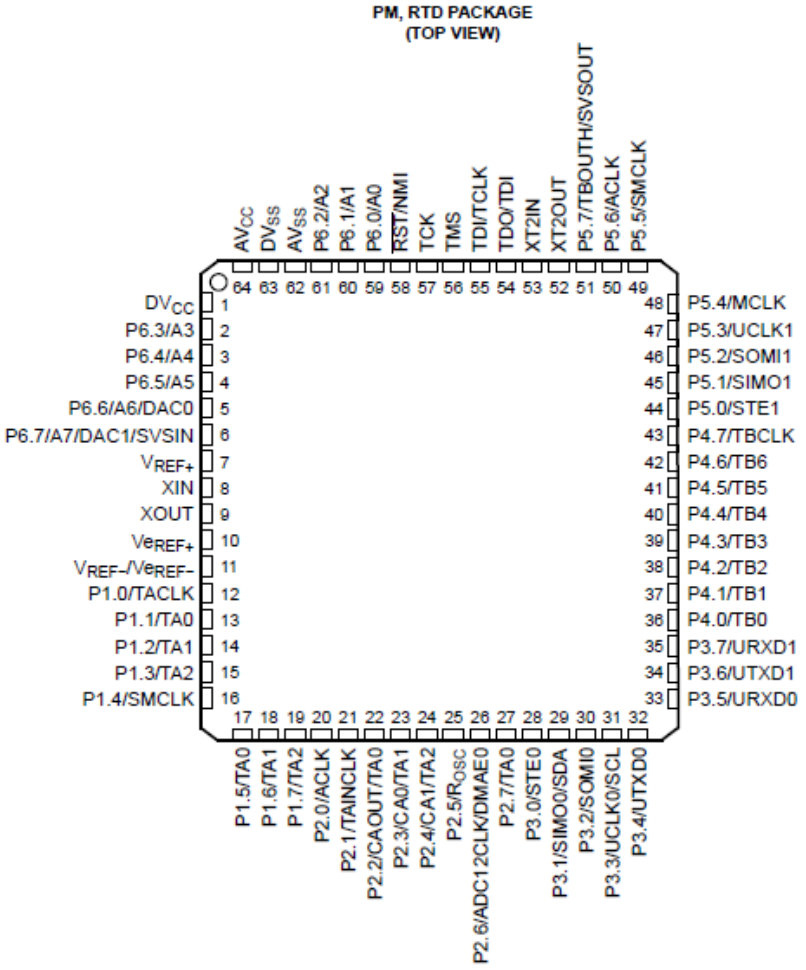


Fig 3.1 : Pin Designation

3.5. UART

A universal asynchronous receiver/transmitter is a piece of computer hardware that translates data between parallel and serial forms. UARTs are commonly used in conjunction with communication standards such as EIA, RS-232, RS-422 or RS-485. The universal designation indicates that the data format and transmission speeds are configurable. The electric signaling levels and methods (such as differential signaling etc.) are handled by a driver circuit external to the UART.

A UART is usually an individual (or part of an) integrated circuit used for serial communications over a computer or peripheral device serial port. UARTs are now commonly included in microcontrollers. A dual UART, or DUART, combines two UARTs into a single chip. An octal UART or OCTART combines eight UARTs into one package, an example being the NXP SCC2698. Many modern ICs now come with a UART that can also communicate synchronously; these devices are called USARTs (universal synchronous/asynchronous receiver/transmitter).

The Universal Asynchronous Receiver/Transmitter (UART) takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes. Each UART contains a shift register, which is the fundamental method of conversion between serial and parallel forms. Serial transmission of digital information (bits) through a single wire or other medium is less costly than parallel transmission through multiple wires.

The UART usually does not directly generate or receive the external signals used between different items of equipment. Separate interface devices are used to convert the logic level signals of the UART to and from the external signalling levels. External signals may be of many different forms. Examples of standards for voltage signaling are RS-232, RS-422 and RS-485 from the EIA. Historically, current (incurrent loops) was used in telegraph circuits. Some signaling schemes do not use electrical wires. Examples of such are optical fiber, IrDA

(infrared), and (wireless) Bluetooth in its Serial Port Profile (SPP). Some signaling schemes use modulation of a carrier signal (with or without wires). Examples are modulation of audio signals with phone line modems, RF modulation with data radios, and the DC-LINfor power line communication.

Character framing

The right-most (least significant) data bit is always transmitted first. If parity is present, the parity bit comes after the data bits but before the stop bit(s).

Bit number	1	2	3	4	5	6	7	8	9	10	11
	Start bit	**5–8 data bits**								**Stop bit(s)**	
	Start	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Stop	

The idle, no data state is high-voltage, or powered. This is a historic legacy from telegraphy, in which the line is held high to show that the line and transmitter are not damaged. Each character is sent as a logic low start bit, a configurable number of data bits (usually 8, but users can choose 5 to 8 or 9 bits depending on which UART is in use), an optional parity bit if the number of bits per character chosen is not 9 bits, and one or more logic high stop bits.

The start bit signals the receiver that a new character is coming. The next five to nine bits, depending on the code set employed, represent the character. If a parity bit is used, it would be placed after all of the data bits. The next one or two bits are always in the **mark** (logic high, i.e., '1') condition and called the stop bit(s). They signal the receiver that the character is completed. Since the start bit is logic low (0) and the stop bit is logic high (1) there are always at least two guaranteed signal changes between characters.

CHAPTER 4
SOFTWARE

4. SOFTWARE

4.1 OPERATING SYSTEM: CONTIKI-2.6

Contiki is open source, highly portable, multi-tasking operating system for memory-efficient networked embedded systems and wireless sensor networks. It has a particular focus on low-power wireless Internet of Things devices. Contiki is designed for microcontrollers with small amounts of memory. A typical Contiki configuration is 2 kilobytes of RAM and 40 kilobytes of ROM. Contiki provides IP communication, both for IPv4 and IPv6.

4.1.1. Features

Contiki supports per-process optional preemptive multi-threading, inter-process communication using message passing through events, as well as an optional GUI subsystem with either direct graphic support for locally connected terminals or networked virtual display with VNC or over Telnet.

A full installation of Contiki includes the following features:

Open Source

Contiki is open source, which means that the source is and always will be available. Contiki may be used in both commercial and non-commercial systems without restrictions.

Low-power operation

Contiki systems are severely power-constrained. Battery operated wireless sensors may need to provide years of unattended operation and with little means to recharge or replace its batteries. Contiki provides a set of mechanisms for reducing the power consumption of the system on which it runs. The default mechanism for attaining low-power operation of the radio is called ContikiMAC. With ContikiMAC, nodes can be running in low-power mode and still be able to receive and relay radio messages.

Sleepy Routers

In wireless networks, nodes may need to relay messages from others to reach their destination. With Contiki, even relay nodes, so-called routers, can be battery-operated thanks to the ContikiMAC radio duty cycling mechanism which allows them to sleep between each relayed message. Some call this "sleeping routers", we call it sleepy routers.

Memory Footprint

Contiki is designed for tiny systems, having only a few kilobytes of memory available. Contiki is therefore highly memory efficient and provides a set of mechanisms for memory allocation: memory block allocation *memb*, a managed memory allocator *mmem*, as well as the standard C memory allocator malloc. A typical system with full IPv6 networking with sleepy routers and RPL routing needs less than 10 k RAM and 30 k ROM.

Hardware

Contiki is designed to run on classes of hardware devices that are severely constrained in terms of memory, power, processing power, and communication bandwidth. A typical Contiki system has memory on the order of kilobytes, a power budget on the order of milliwatts, processing speed measured in megahertz, and communication bandwidth on the order of hundreds of kilobits/second. This class of systems includes both various types of embedded systems as well as a number of old 8-bit computers.

Full IP Networking

Contiki provides a full IP network stack, with standard IP protocols such as UDP, TCP, and HTTP, in addition to the new low-power standards like 6lowpan, RPL, and CoAP.

Contiki provides three network mechanisms: the uIP TCP/IP stack,which provides IPv4 networking, the uIPv6 stack,[6] which provides IPv6 networking, and the Rime stack, which is a set of custom lightweight networking protocols designed specifically for low-power wireless networks. The IPv6 stack was contributed by Cisco and was, at the time of release, the smallest IPv6 stack to receive the IPv6 Ready certification.The IPv6 stack also

contains the RPL routing protocol for low-power lossy IPv6 networks and the 6LoWPAN header compression and adaptation layer for IEEE 802.15.4 links.

The Rime stack is an alternative network stack that is intended to be used when the overhead of the IPv4 or IPv6 stacks is prohibitive. The Rime stack provides a set of communication primitives for low-power wireless systems. The default primitives are single-hop unicast, single-hop broadcast, multi-hop unicast, network flooding, and address-free data collection. The primitives can be used on their own or combined to form more complex protocols and mechanisms.

Internet Standards

Contiki provides powerful low-power Internet communication. Contiki supports fully standard IPv6, IPv4 and HTTP, along with the recent low-power wireless standards: 6lowpan, RPL, CoAP. With Contiki's ContikiMAC and sleepy routers, even wireless routers can be battery-operated.

6lowPAN, RPL, CoAP

Contiki supports the recently standardized IETF protocols for low-power IPv6 networking, including the 6lowpan adaptation layer, the RPL IPv6 multi-hop routing protocol, and the CoAP RESTful application-layer protocol.

The Rime Stack

In situations when bandwidth is at a premium or where the full IPv6 networking stack is overkill, Contiki provides a tailored wireless networking stack called Rime. The Rime stack supports simple operations such as sending a message to all neighbors or to a specified neighbor, as well as more complex mechanisms such as network flooding and address-free multi-hop semi-reliable scalable data collection. Everything runs with sleepy routers to save power.

Programming model

To save memory but provide a nice control flow in the code, Contiki uses a mechanism called protothreads. Protothreads is a mixture of the event-driven and the multi-threaded programming mechanisms. With protothreads, event-handlers can be made to block, waiting for events to occur. A protothread is a memory-efficient programming abstraction that shares features of both multi-threading and event-driven programming to attain a low memory overhead of each protothread. The kernel invokes the protothread of a process in response to an internal or external event. Examples of internal events are timers that fire or messages being posted from other processes. Examples of external events are sensors that trigger or incoming packets from a radio neighbor.

Protothreads are cooperatively scheduled. This means that a Contiki process must always explicitly yield control back to the kernel at regular intervals. Contiki processes may use a special protothread construct to block waiting for events while yielding control to the kernel between each event invocation.

Simulation

The Contiki system includes a network simulator called Cooja.Contiki devices often make up large wireless networks. Developing and debugging software for such networks is really hard. Cooja, the Contiki network simulator, makes this tremendously easier by providing a simulation environment that allows developers to both see their applications run in large-scale networks or in extreme detail on fully emulated hardware devices. Cooja simulates networks of Contiki nodes. The nodes may belong to either of three classes: emulated nodes, where the entire hardware of each node is emulated, Cooja nodes, where the Contiki code for the node is compiled for and executed on the simulation host, or Java nodes, where the behavior of the node must be re-implemented as a Java class. A single Cooja simulation may contain a mixture of nodes from any of the three classes. Emulated nodes can also be used to include non-Contiki nodes in a simulated network.

In Contiki 2.6, platforms with the TI MSP430 and Atmel AVR microcontrollers can be emulated.

4.2.2. Contiki Erbium CoAP

The Constrained Application Protocol (CoAP) is an application layer protocol (IETF draft) for resource constrained devices.

It is a specialized web transfer protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks. The nodes often have 8-bit microcontrollers with small amounts of ROM and RAM, while constrained networks such as 6LoWPAN often have high packet error rates and a typical throughput of 10s of kbit/s. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation.

CoAP provides a request/response interaction model between application endpoints, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs and Internet media types. CoAP is designed to easily interface with HTTP for integration with the Web while meeting specialized requirements such as multicast support, very low overhead and simplicity for constrained environments.

The use of web services (web APIs) on the Internet has become ubiquitous in most applications, and depends on the fundamental Representational State Transfer [REST] architecture of the web.

The Constrained RESTful Environments (CoRE) work aims at realizing the REST architecture in a suitable form for the most constrained nodes and networks. Constrained networks such as 6LoWPAN support the fragmentation of IPv6 packets into small link-layer frames, however incurring significant reduction in packet delivery probability. One design goal of CoAP has been to keep message overhead small, thus limiting the need for fragmentation.

One of the main goals of CoAP is to design a generic web protocol for the special requirements of this constrained environment, especially considering energy, building automation and other

machine-to-machine (M2M) applications. The goal of CoAP is not to blindly compress HTTP [RFC2616], but rather to realize a subset of REST common with HTTP but optimized for M2M applications. Although CoAP could be used for refashioning simple HTTP interfaces into a more compact protocol, it more importantly also offers features for M2M such as built-in discovery, multicast support and asynchronous message exchanges.

4.3. CoAP

The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks. The nodes often have 8-bit microcontrollers with small amounts of ROM and RAM, while constrained networks such as 6LoWPAN often have high packet error rates and a typical throughput of 10s of Kbit/s. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation.

CoAP provides a request/response interaction model between application endpoints, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs and Internet media types. CoAP is designed to easily interface with HTTP for integration with the Web while meeting specialized requirements such as multicast support, very low overhead and simplicity for constrained environments.

The use of web services (web APIs) on the Internet has become ubiquitous in most applications, and depends on the fundamental Representational State Transfer [REST] architecture of the web.

The Constrained RESTful Environments (CoRE) work aims at realizing the REST architecture in a suitable form for the most constrained nodes (e.g. 8-bit microcontrollers with limited RAM and ROM) and networks (e.g. 6LoWPAN, [RFC4944]). Constrained networks such as 6LoWPAN support the fragmentation of IPv6 packets into small link-layer frames, however incurring significant reduction in packet delivery probability. One design goal of CoAP has been to keep message overhead small, thus limiting the need for fragmentation.

One of the main goals of CoAP is to design a generic web protocol for the special requirements of this constrained environment, especially considering energy, building automation and other machine-to-machine (M2M) applications. The goal of CoAP is not to blindly compress HTTP [RFC2616], but rather to realize a subset of REST common with HTTP but optimized for M2M applications. Although CoAP could be used for refashioning simple HTTP interfaces into a more compact protocol, it more importantly also offers features for M2M such as built-in discovery,

multicast support and asynchronous message exchanges.

4.3.1. FEATURES

CoAP has the following main features:

- Constrained web protocol fulfilling M2M requirements.
- UDP [RFC0768] binding with optional reliability supporting unicast and multicast requests.
- Asynchronous message exchanges.
- Low header overhead and parsing complexity.
- URI and Content-type support.
- Simple proxy and caching capabilities.
- A stateless HTTP mapping, allowing proxies to be built providing access to CoAP resources via HTTP in a uniform way or for HTTP
- Simple interfaces to be realized alternatively over CoAP.
- Security binding to Datagram Transport Layer Security (DTLS)

The interaction model of CoAP is similar to the client/server model of HTTP. However, machine-to-machine interactions typically result in a CoAP implementation acting in both client and server roles. A CoAP request is equivalent to that of HTTP, and is sent by a client to request an action (using a method code) on a resource (identified by a URI) on a server. The server then sends a response with a response code; this response may include a resource representation.

Unlike HTTP, CoAP deals with these interchanges asynchronously over datagram-oriented transport such as UDP. This is done logically using a layer of messages that supports optional reliability (with exponential back-off). CoAP defines four types of messages:
Confirmable, Non-confirmable, Acknowledgement, Reset; method codes and response codes included in some of these messages make them carry requests or responses.

The basic exchanges of the four types of messages are somewhat orthogonal to the request/response interactions; requests can be carried in Confirmable and Non-confirmable messages, and responses can be carried in these as well as piggy-backed in Acknowledgement messages.

One could think of CoAP logically as using a two-layer approach, a CoAP messaging layer used to deal with UDP and the asynchronous nature of the interactions, and the request/response interactions using Method and Response codes (see Figure 3) CoAP is however a single protocol, with messaging and request/response just features of the CoAP header.

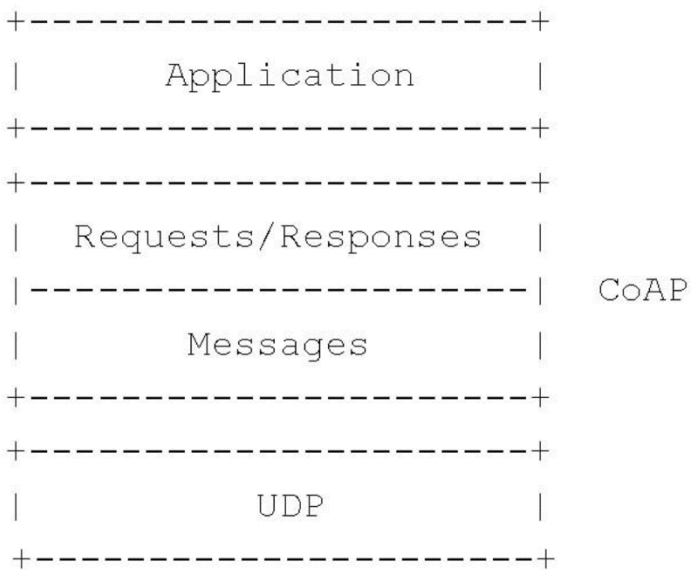


Figure 7: Abstract layering of CoAP

4.3.2. Message Format

CoAP is based on the exchange of compact messages which, by default, are transported over UDP (i.e. Each CoAP message occupies the data section of one UDP datagram). CoAP may also be used over Datagram Transport Layer Security (DTLS). It could also be used over other transports such as SMS, TCP or SCTP, the specification of which is out of this document's scope. (UDP-lite and UDP zero checksum are not supported by CoAP.)

CoAP messages are encoded in a simple binary format. The message format starts with a fixed-size 4-byte header. This is followed by a variable-length Token value which can be between 0 and 8 bytes long. Following the Token value comes a sequence of zero or more CoAP. Options in Type-Length-Value (TLV) format, optionally followed by a payload which takes up the rest of the datagram.

CoAP messages are exchanged asynchronously between CoAP endpoints. They are used to transport CoAP requests and responses. As CoAP is bound to non-reliable transports such as UDP, CoAP messages may arrive out of order, appear duplicated, or go missing without notice. For this reason, CoAP implements a lightweight reliability mechanism, without trying to re-create the full feature set of a transport like TCP. It has the following features:

- Simple stop-and-wait retransmission reliability with exponential-off for Confirmable messages.

- Duplicate detection for both Confirmable and Non-confirmable messages.

4.3.3. METHOD DEFINITIONS

GET

The GET method retrieves a representation for the information that currently corresponds to the resource identified by the request URI. If the request includes an Accept Option, that indicates the preferred content-format of a response. The GET method is safe and idempotent.

POST

The POST method requests that the representation enclosed in the request be processed. The actual function performed by the POST method is determined by the origin server and dependent on the target resource. It usually results in a new resource being created or the target resource being updated. POST is neither safe nor idempotent.

PUT

The PUT method requests that the resource identified by the request URI be updated or created with the enclosed representation. The representation format is specified by the media type and content coding given in the Content-Format Option, if provided. PUT is not safe, but is idempotent.

DELETE

The DELETE method requests that the resource identified by the request URI be deleted. DELETE is not safe, but is idempotent.

4.4. SLIP

The Serial Line Internet Protocol (SLIP) is an encapsulation of the Internet Protocol designed to operate through a serial connection. SLIP is a very simple protocol that frames IP datagrams to send them over serial connections. Although it is mostly obsolete now, thanks to its small overhead, it is still used for connecting constrained embedded syste

4.5. Constrained RESTful Environment (CoRE) Architecture

The Constrained Application Protocol (CoAP) adheres to RESTful approach for managing resources and supports mapping to HTTP for Web integration. CoAP resources are identified by Uniform Resource Identifiers (URI).

The Constrained RESTful Environments (CoRE) working group aims at realizing the REST architecture in a suitable form for the most constrained nodes (e.g. 8-bit microcontrollers with limited RAM and ROM) and networks (e.g. 6LoWPAN). CoRE is aimed at machine-to-machine (M2M) applications such as smart energy and building automation.

REST architectures allow IoT and Machine-to-Machine (M2M) applications to be developed on top of web services which can be shared and reused. The sensors become abstract resources identified by URIs, represented with arbitrary formats and manipulated with the same methods as HTTP. As a consequence, RESTful WSNs drastically reduce the application development complexity. The use of web service in LLNs is not straightforward as a consequence of the differences between Internet applications and IoT or M2M applications. IoT or M2M applications are short-lived and web services reside in battery operated devices which most of the time sleep and wakeup only when there is data traffic to be exchanged. In addition, such applications require a multicast and asynchronous communication compared to the unicast and synchronous approach of standard Internet applications .

The Internet Engineering Task Force (IETF) Constrained RESTful environments (CoRE) Working Group has done major standardization work for introducing the web service paradigm into networks of smart objects. The CoRE group has defined a REST based web transfer protocol called Constrained Application Protocol (CoAP). CoAP includes the HTTP functionalities which have been re-designed taking into account the low processing power and energy consumption constraints of small embedded devices such as sensors. In order to make the protocol suitable to IoT and M2M applications, various new functionalities have been added. The major differences between CoAP and HTTP and compares the two protocols in terms of power consumption and

overhead. The power consumption is drastically lower when using CoAP compared to HTTP. The application allows a user to access WSN data directly from a Web browser.

4.6. Web Implementation of CoAP - Cu Plug In

'Copper,' a generic browser for the Internet of Things based on the Constrained Application Protocol (CoAP). Current estimates foresee that the number of networked embedded devices encompassed by the Internet of Things will be vast. Additionally, most systems will be optimized for the constrained environment with its limited energy and bandwidth. These factors make it difficult for users to observe and control the devices. Thus, a major problem will be node and network management, as experienced before in large wireless sensor network deployments. By adopting well-known patterns from the Web, such as browsing, bookmarking, and linking, we provide a user-friendly management tool for networked embedded devices. By integrating it into the Web browser, we allow for intuitive interaction and a presentation layer that is originally missing in the CoAP protocol suite.

Features:

- URI handling for the 'CoAP' scheme (address bar and links)
- Interaction through GET, POST, PUT, and DELETE
- Resource discovery
- Blockwise transfers
- Observing Resources

4.6.1. URI handling for the 'CoAP' scheme(address bar and links)

CoAP URI Scheme

coap-URI = "coap:" "//" host [":" port] path-abempty ["?" query]

If the host component is provided as an IPv4 address or IPv6address, then the CoAP server can be reached at that IP address. If host is a registered name, then that name is considered an indirect identifier and the endpoint might use a name resolution service, such as DNS, to find the address of that host. The host MUST NOT be empty; if a URI is received with a missing authority or an empty host, then it MUST be considered invalid. The port subcomponent indicates the UDP port at which the CoAP server is located. If it is empty or not given, then the default port 5683 is assumed.

Application designers are encouraged to make use of short, but descriptive URIs. As the environments that CoAP is used in are usually constrained for bandwidth and energy, the trade-off between these two qualities should lean towards the shortness, without ignoring descriptiveness.

4.6.2. Resource Discovery

Resource Discovery can be performed either unicast or multicast. When a server's IP address is already known, either a priori or resolved via the Domain Name System (DNS), unicast discovery is performed in order to locate the entry point to the resource of interest. In this specification, this is performed using a GET to the server, which returns a payload in the CoRE Link Format. A client would then match the appropriate Resource Type, Interface Description and possible Media type for its application. These attributes may also be included in the query string in order to filter the number of links returned in a response.

Multicast resource discovery is useful when a client needs to locate a resource within a limited scope, and that scope supports IP multicast. A GET request to the appropriate multicast address is made. In order to limit the number and size or responses, a query string is recommended with the known attributes. Typically a resource would be discovered based on its Resource Type and/or Interface Description, along with possible application specific attributes.

4.6.3. Modelling Soil Sensor Properties as CoAP Resource

CoAP resource parameters - resource name, methods supported, URI Path string, resource type. Each resource has to implement an associated handler function CoAP response can be plain text, xml, JSON based on client requested format.

Example:

1. RESOURCE(temperature, METHOD GET, "onboard-sensors/temperature", "title="Sensirion Temperature Sensor(supports JSON)";rt="TemperatureSensor"");
2. RESOURCE(vwc, METHOD GET, "soil-sensors/VWC", "title="5TE Soil Sensor (supportsJSON)";rt="SoilSensor"");

4.7. IPv6

Internet Protocol version 6 (IPv6) is the latest version of the Internet Protocol (IP), the communications protocol that provides an identification and location system for computers on networks and routes traffic across the Internet. IPv6 was developed by the Internet Engineering Task Force (IETF) to deal with the long-anticipated problem of IPv4 address exhaustion.

IPv6 is intended to replace IPv4, which still carries more than 96% of Internet traffic worldwide as of May 2014. As of February 2014, the percentage of users reaching Google services over IPv6 surpassed 3% for the first time.

Every device on the Internet is assigned an IP address for identification and location definition. With the ever-increasing number of new devices being connected to the Internet, the need arose for more addresses than the IPv4 address space has available. IPv6 uses a 128-bit address, allowing 2^{128}, or approximately 3.4×10^{38} addresses, or more than 7.9×10^{28} times as many as IPv4, which uses 32-bit addresses. IPv4 provides approximately 4.3 billion addresses. The two protocols are not designed to be interoperable, complicating the transition to IPv6.

IPv6 addresses are represented as eight groups of four hexadecimal digits separated by colons, for example 2001:0db8:85a3:0042:1000:8a2e:0370:7334, but methods of abbreviation of this full notation exist.

4.7.1 Benefits of IPv6

With IPv6, everything from appliances to automobiles can be interconnected. But an increased number of IT addresses isn't the only advantage of IPv6 over IPv4. With IPv6, everything from appliances to automobiles can be interconnected. But an increased number of IT addresses isn't the only advantage of IPv6 over IPv4

- **More Efficient Routing**

  IPv6 reduces the size of routing tables and makes routing more efficient and hierarchical. IPv6 allows ISPs to aggregate the prefixes of their customers' networks into a single prefix and announce this one prefix to the IPv6 Internet. In addition, in IPv6 networks, fragmentation is handled by the source device, rather than the router, using a protocol for discovery of the path's maximum transmission unit (MTU).

- **More Efficient Packet Processing**

  IPv6's simplified packet header makes packet processing more efficient. Compared with IPv4, IPv6 contains no IP-level checksum, so the checksum does not need to be recalculated at every router hop. Getting rid of the IP-level checksum was possible because most link-layer technologies already contain checksum and error-control capabilities. In addition, most transport layers, which handle end-to-end connectivity, have a checksum that enables error detection.

- **Directed Data Flows**

  IPv6 supports multicast rather than broadcast. Multicast allows bandwidth-intensive packet flows (like multimedia streams) to be sent to multiple destinations simultaneously, saving network bandwidth. Disinterested hosts no longer must process broadcast packets. In addition, the IPv6 header has a new field, named Flow Label that can identify packets belonging to the same flow.

- **Simplified Network Configuration**

  Address auto-configuration (address assignment) is built in to IPv6. A router will send the prefix of the local link in its router advertisements. A host can generate its own IP address by appending its link-layer (MAC) address, converted into Extended Universal Identifier (EUI) 64-bit format, to the 64 bits of the local link prefix.

- **Support for New Services**

  By eliminating Network Address Translation (NAT), true end-to-end connectivity at the IP layer is restored, enabling new and valuable services. Peer-to-peer

networks are easier to create and maintain, and services such as VoIP and Quality of Service (QoS) become more robust.

- **Security**

  IPSec, which provides confidentiality, authentication and data integrity, is baked into in IPv6. Because of their potential to carry malware, IPv4 ICMP packets are often blocked by corporate firewalls, but ICMPv6, the implementation of the Internet Control Message Protocol for IPv6, may be permitted because IPSec can be applied to the ICMPv6 packets.

6LoWPAN is an acronym of *IPv6 over Low power Wireless Personal Area Networks*.6LoWPAN is the name of a concluded working group in the Internet area of the IETF.

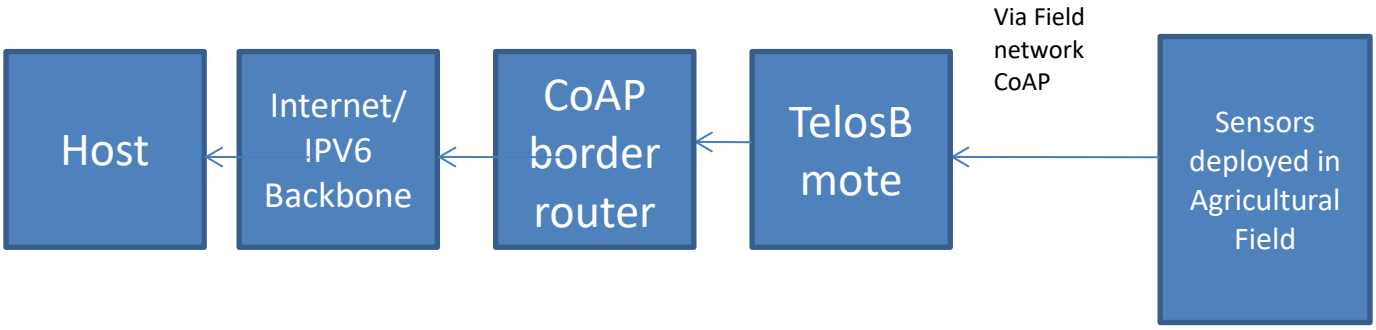
The 6LoWPAN concept originated from the idea that "the Internet Protocol could and should be applied even to the smallest devices,"and that low-power devices with limited processing capabilities should be able to participate in the Internet of Things.

CHAPTER 5

IMPLEMENTATION

5. IMPLEMENTATION

5.1. BLOCK DIAGRAM



The block diagram shows the monitoring of remote agricultural sensor resources using CoAP. The architecture has two network segments, namely ***Agricultural field network*** and ***Monitoring Network***. Field network consist of motes that are interfaced to soil sensors.

The motes run CoAP server managing the monitorable CoAP resources. The soil sensor resources and onboard sensor resources can be discovered and CoAP methods can be acted on them using the identified CoAP URI. The CoAP application network will be connected to the IPv6 backbone/internet using the CoAP Proxy/border router. The CoAP proxy is the gateway for the field network to connect internet using cellular network.

Monitoring network consists of CoAP client node(s) and web server. CoAP clients can be utilized for real-time access to sensor data, while web server can give access archived data to the farmer and the agricultural scientist.

Shown below is the deployment architecture for agriculture usecase. The ***Monitoring network*** is on the left side consisting of the user and the Computer. The ***Agricultural field network*** is present on the right side consisting of the motes and the sensors. The motes are connected to the soil sensors

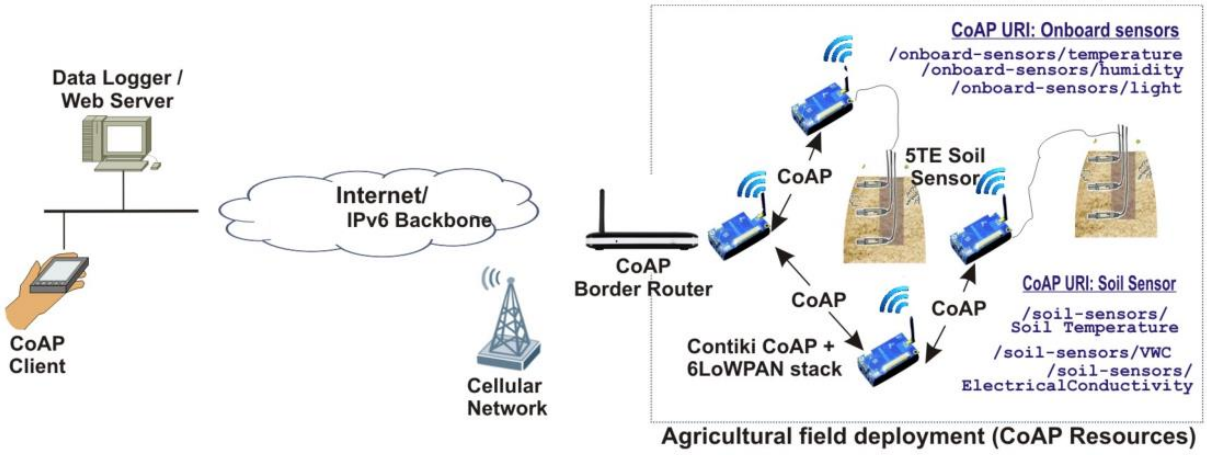


Fig. 4. Deployment Architecture for Agriculture Usecase

Figure 8: Deployment Architecture for Agriculture Usecase

CHAPTER 6

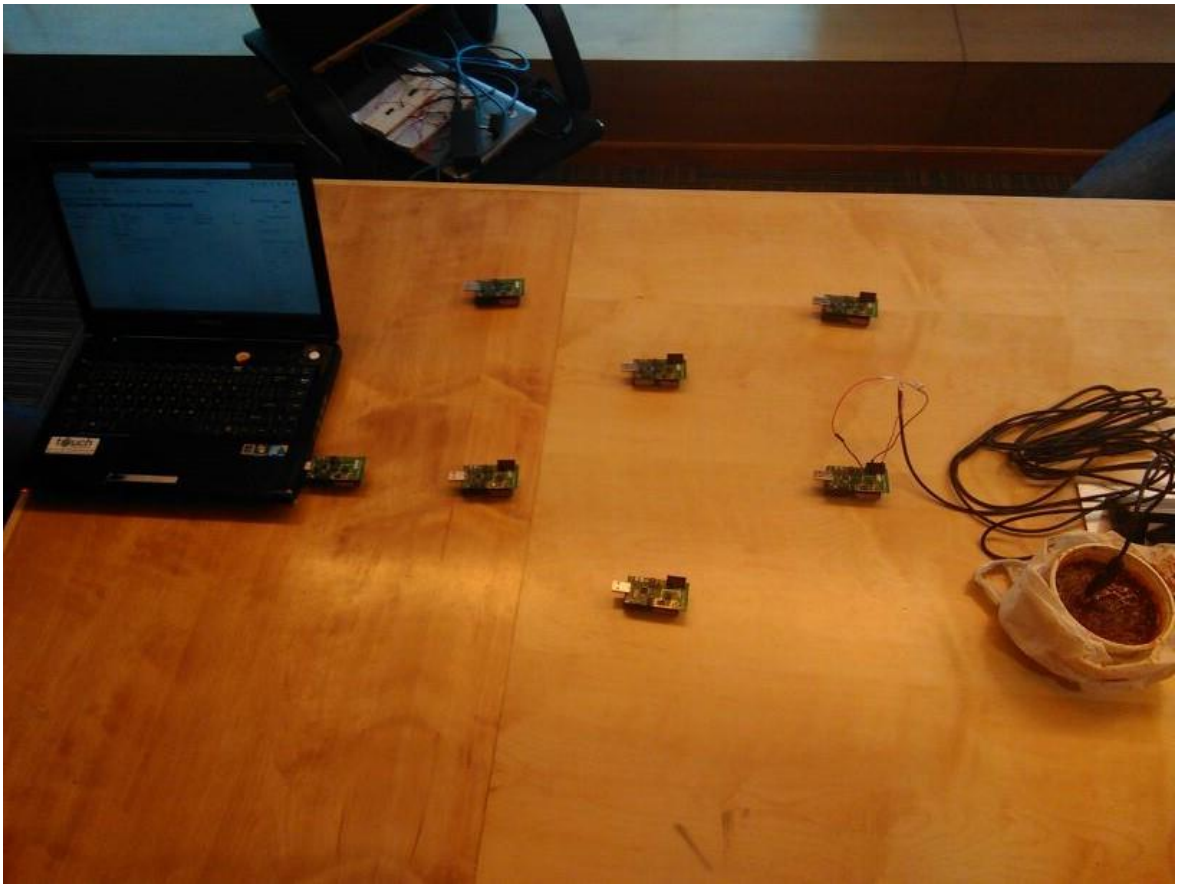
RESULTS

6. RESULTS

A Wireless Sensor network is established by setting up the TelosB motes at a distance away from each other. The sensors are connected to the Motes. The sensors require a trigger through which the sensors will be able to measure the soil parameters. The trigger of 3V will be supplied by the motes to the sensor.

On pressing the GET button in the Copper Add-on in the Mozilla Firefox Window, the 5TE soil sensor will be triggered by the Mote by a value of 3V. The sensor will measure the soil parameters and the data will be sent by the sensor to the RX pin of the mote. UART mode of communication occurs and the mote will receive the sensor data through the RX pin of the mote.

The data will be received serially. The received data is in ASCII value which needs to suitably converted into a decimal value to obtain the soil parameters in a recognizable format.



APPLICATIONS

- Wireless sensor networks can be used in precision farming, the sensors are deployed over a region where soil parameters is monitored.

- Wireless sensor networks have been deployed to monitor the concentration of dangerous gases for citizens. These can take the advantage of the ad hoc wireless links rather than wired installations, which also make them more mobile for testing readings in different areas.

- Water properties in dams, rivers, lakes and oceans, as well as underground water reserves are monitored. The use of many wireless distributed sensors enables the creation of a more accurate map of the water status, and allows the permanent deployment of monitoring stations in locations of difficult access, without the need of manual data retrieval.

- Collection of data for monitoring of environmental information can be done. This can be as simple as the monitoring of the temperature in a fridge to the level of water in overflow tanks in nuclear power plants.

- Wireless sensor networks can effectively act to prevent the consequences of natural disasters, like floods. Wireless nodes have successfully been deployed in rivers where changes of the water levels have to be monitored in real time.

CHAPTER 7
CONCLUSION AND FUTURE SCOPE OF WORK

- The number of devices that will connect to the Future Internet is increasing exponentially. The 6LoWPAN adaptation layer enables assignment of IPv6 addresses to low-power wireless devices making them reachable from any other node on the internet. In this report, CoAP open application layer protocol utilized for real-time monitoring of IP-enabled agriculture sensors network is observed.

- Future work includes the field deployment of CoAP-based agriculture sensors network and its connectivity to IPv6 backbone for real-time monitoring over the internet. This project can be used in precision architecture where even minute details of soil are important, and has to be accounted to yield desired output. Precision agriculture provides farmers with a wealth of information to improve decision making and enhance the inherent quality of farm products.

- Further, more agriculture sensors such as soil pH and leaf wetness can be used to improve the efficiency. In addition to monitoring, motes can be connected to actuators in order to effect irrigation in response to a particular condition. Water can be sprinkled automatically if the acidity levels in the soil increases. This reduces the dependability of labour in farming.

- The concept of Internet of Things (IOT) can be implemented for a smarter world. Internet of Things is an advanced connectivity of devices, systems and services that goes beyond the traditional machine-to-machine and covers a variety of protocols, domains and applications. The next generation of Internet applications using IPV6 would be able to communicate with devices attached to virtually all human-made objects because of the extremely large address space of the IPV6 protocol. This system would therefore be able to scale to the large number of objects envisaged. A person's ability to interact with objects could be altered remotely based on immediate or present needs, in accordance

with end-user agreements. Such technology could enable much more powerful control of content creators and owners over their creations by better applying copyright restrictions and digital restrictions management, so a customer buying a Blu-ray disc containing a movie could choose to pay a high price and be able to watch the movie for a whole year, pay a moderate price and have the right to watch the movie for a week, or pay a low fee every time she or he watches the movie.

APPENDIX

SOIL PARAMETERS RECEIVE SERVER CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "contiki.h"
#include "contiki-net.h"
#include "isr_compat.h"
#include "msp430.h"
#include "dev/watchdog.h"
void delay(long int);                          //Global Declarations
void getdata();
void deinit();
void uartinit();

#if !UIP_CONF_IPV6_RPL && !defined (CONTIKI_TARGET_MINIMAL_NET) &&
!defined (CONTIKI_TARGET_NATIVE)
#warning "Compiling with static routing!"
#include "static-routing.h"
#endif

#include "erbium.h"


#if defined (PLATFORM_HAS_BUTTON)       /*Declarations Defining the
components of the sensor node(mote) such as Button,leds,lights,battery and
radio*/
#include "dev/button-sensor.h"
#endif
#if defined (PLATFORM_HAS_LEDS)
#include "dev/leds.h"
#endif
#if defined (PLATFORM_HAS_LIGHT)
#include "dev/light-sensor.h"
#endif
#if defined (PLATFORM_HAS_BATTERY)
#include "dev/battery-sensor.h"
#endif
#if defined (PLATFORM_HAS_SHT11)
#include "dev/sht11-sensor.h"
#endif
#if defined (PLATFORM_HAS_RADIO)
#include "dev/radio-sensor.h"
#endif


/* For CoAP-specific example: not required for normal RESTful Web service.
Defining that coap 08 is compatible with this code*/
```

```
#if WITH_COAP == 3
#include "er-coap-03.h"
#elif WITH_COAP == 7
#include "er-coap-07.h"
#else
#warning "Erbium example without CoAP-specifc functionality"
#endif /* CoAP-specific example */

#define DEBUG 0        //Code needed for using printf statements
#if DEBUG
#define PRINTF(...) printf(__VA_ARGS__)
#define PRINT6ADDR(addr)
PRINTF("[%02x%02x:%02x%02x:%02x%02x:%02x%02x:%02x%02x:%02x%02x:%02x%02x:%0
2x%02x]", ((uint8_t *)addr)[0], ((uint8_t *)addr)[1], ((uint8_t
*)addr)[2], ((uint8_t *)addr)[3], ((uint8_t *)addr)[4], ((uint8_t
*)addr)[5], ((uint8_t *)addr)[6], ((uint8_t *)addr)[7], ((uint8_t
*)addr)[8], ((uint8_t *)addr)[9], ((uint8_t *)addr)[10], ((uint8_t
*)addr)[11], ((uint8_t *)addr)[12], ((uint8_t *)addr)[13], ((uint8_t
*)addr)[14], ((uint8_t *)addr)[15])
#define PRINTLLADDR(lladdr)
PRINTF("[%02x:%02x:%02x:%02x:%02x:%02x]",(lladdr)->addr[0], (lladdr)-
>addr[1], (lladdr)->addr[2], (lladdr)->addr[3],(lladdr)->addr[4],
(lladdr)->addr[5])
#else
#define PRINTF(...)
#define PRINT6ADDR(addr)
#define PRINTLLADDR(addr)
#endif

char dest[30];
int de = 0, ec = 0, tp =0,dec=0,edec=0,dep=0,det=0;// declaring the soil
parameter variables
char msg[11]="55 128 600";
static int l;
/* A simple actuator example. Toggles the red led */
RESOURCE(toggle, METHOD_GET , "receive", "title=\"Red
LED\";rt=\"Control\"");           //Receive resource defined here
void
toggle_handler(void* request, void* response, uint8_t *buffer, uint16_t
preferred_size, int32_t *offset)
{

      printf("entered resource\n");
      uartinit();

      P2OUT |= BIT3;                    //Selection of a GPIO pin
      l=0;

      delay(150000);                    // A delay is given for
                                  triggering the sensor
      P2OUT ^= BIT3;                    //Excitation of a GPIO pin
      printf("%s\n",dest);
      getdata();
```

```
      deinit();

      REST.set_header_content_type(response, REST.type.APPLICATION_XML);
      //To Display the data in the Copper Add-on in the Mozilla Firefox
Web Browser
        snprintf((char *)buffer, REST_MAX_CHUNK_SIZE, "Dielectric=
%d.%d\nElectrical conductivity= %d.%d\nTemperature=
%d.%d\n",de,dep,ec,edec,tp,dec);

        REST.set_response_payload(response, buffer, strlen((char
*)buffer));

      //delay(100000);
      printf("exit resource\n");

}


PROCESS(rest_server_example, "Erbium Example Server");
AUTOSTART_PROCESSES(&rest_server_example);


PROCESS_THREAD(rest_server_example, ev, data)
{
  PROCESS_BEGIN();

  PRINTF("Starting Erbium Example Server\n");

#ifdef RF_CHANNEL
  PRINTF("RF channel: %u\n", RF_CHANNEL);
#endif
#ifdef IEEE802154_PANID
  PRINTF("PAN ID: 0x%04X\n", IEEE802154_PANID);
#endif

  PRINTF("uIP buffer: %u\n", UIP_BUFSIZE);
  PRINTF("LL header: %u\n", UIP_LLH_LEN);
  PRINTF("IP+UDP header: %u\n", UIP_IPUDPH_LEN);
  PRINTF("REST max chunk: %u\n", REST_MAX_CHUNK_SIZE);

/* if static routes are used rather than RPL */
#if !UIP_CONF_IPV6_RPL && !defined (CONTIKI_TARGET_MINIMAL_NET) &&
!defined (CONTIKI_TARGET_NATIVE)
  set_global_address();
  configure_routing();
#endif

  /* Initialize the REST engine. */
  rest_init_engine();



      rest_activate_resource(&resource_toggle);     //Activate the
resource
```

```
  while(1) {
printf("enters while\n");
    PROCESS_WAIT_EVENT();
printf("repeat while\n");
      } /* while (1) */

  PROCESS_END();
}

ISR(UART0RX,uart0_rx_interrupt)         //Interrupt Service Routine
{

      //printf("%c",RXBUF0);
      if(l<15)
      {
      dest[l-1] = RXBUF0;
l++;
}
IFG1 &= ~URXIFG0;
return;

}

void uartinit()
{
      WDTCTL = WDTPW + WDTHOLD;
      P3SEL |= 0x20;                    //Select UART0 Transmit and
Receive
      P3DIR &= ~0x20;                         //Make URXD0 as input


      ME1 |= URXE0;                     //module enable
      U0CTL = CHAR;                     //Select 8 bit transceive mode
      U0TCTL = 147;                     //ACLK = UCLK
      //U0RCTL = 0x00;
      U0BR0=0x1b;                  //Baud rate register
      U0BR1=0x00;
      U0MCTL=0x24;                      //Modulation control register
      U0CTL &= ~SWRST;             //UART Software Reset

      IE1 = URXIE0;                     //Enable Receive Interrrupt
      P2SEL &= (~BIT3);            //Select Port 2
      P2DIR |= BIT3;
}

void delay(long int n)
{
      long int i;
      for(i=0;i<n;i++)
      _NOP();
}
```

```
void getdata()                          //Function for receiving Sensor
Data and converting ASCII to Decimal
{

      int k=0;
      de = 0;
      tp = 0;
      ec = 0;

      while(dest[k]!=32)
      {
            de = de*10 + (dest[k] - 48);
            k++;
      }
      k++;
      while(dest[k]!=32)
      {
            ec = ec*10 + (dest[k] - 48);
            k++;
      }
      k++;
      while(dest[k]!=0x0D)
      {
            tp = tp*10 + (dest[k] - 48);
            k++;
      }

      printf("%d,%d,%d\n",de,ec,tp);
      det=de % 50;
      dep=(det * 10)/50;
      de = de/50;
      edec=ec%100;
      ec = ec/100;
      dec = tp % 10;
      tp = (tp-400)/10;
      printf("%d,%d,%d.%d\n",de,ec,tp,dec);
      P2SEL &= (~BIT6);
      P2DIR |= BIT6;
      if(tp>28)
      {
      P2OUT |=BIT6;
      }
      else
      P2OUT = 0 ;
}

void deinit()                       //De-initialization which helps in
Displaying the data on the Copper Add on in Mozilla Firefox Browser
{
      U0BR0=0x02;

        U0BR1=0x00;
```

```
        U0MCTL=0x00;


      IFG1 = 130;




      U0TCTL = 163;
      //U0RCTL = 0;
      IE1 = WDTIE;
      U0CTL = 22;
}
```

REFERENCES

[1] Disruptive Civil Technologies: Six Technologies with Potential Impacts on US Interests out to 2025. National Intelligence Council, April 2008.

[2] Internet of Things An action plan for Europe. http://ec.europa.eu/information society/policy/rfid/documents/commiot2009.pdf.

[3] Towards 50 Billion Devices: Personal & M2M. http://www.ericsson.com/campaign/opportunitysupportsystems/newsfeed/posts/15-heading-towards-50-billion-connections/.

[4] D. Evans, Top 25 Technology Predictions. Cisco IBSG Innovations Practice, 2009.

[5] IEEE Std 802.15.4 Specifications for Low-Rate Wireless Personal Area Networks WPANs. IEEE Computer Society, September 2006.

[6] G. Montenegro, N. Kushalnagar, J. Hui and D. Culler, Transmission of IPv6 Packets over IEEE 802.15.4 Networks. IETF RFC 4944, September 2007.

[7] J. Hui and P. Thubert, Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. IETF RFC 6282, September 2011.

[8] Z. Shelby, K. Hartke, C. Bormann and B. Frank, Constrained Application Protocol (CoAP). draft-ietf-core-coap-11, July, 2012.

[9] Constrained RESTful Environments (core) Working Group. http://datatracker.ietf.org/wg/core/.

[10] C. Bormann, A. Castellani and Z. Shelby, CoAP: An Application Protocol for Billions of Tiny Internet Nodes. IEEE Internet Computing, 16(2):62-67, 2012

[11] R. Fielding, ”Architectural Styles and the Design of Network-based Software Architectures. Ph.D. Dissertation, University of California, Irvine, 2000.

[12] TNAU agri portal - precision farming. http://agritech.tnau.ac.in/pres farm agri.html.

[13] 5TE Soil Sensor. http://www.decagon.com/products/sensors/soil-

moisture-sensors/, 2012

[14] Contiki: The Operating System for Connecting the Next Billion Devices - the Internet of Things. http://www.sics.se/node/108, 2012.

[15] A. Dunkels, J. Alonso and T. Voigt, Making TCP/IP Viable for Wireless Sensor Networks. In First European Workshop on Wireless Sensor Networks, 2004.

[16] T. Winter and P. Thubert (Eds), RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. IETF RFC 6550, March 2012.

[17] M. Kovatsch, S. Duquennoy and A.Dunkels, A Low-Power COAP for Contiki. In IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems, 2011.

[18] D. Harrington, R. Presuhn and B. Wijnen, An Architecture for Describing SNMP Management Frameworks. IETF RFC 3411, December 2002.

[19] 6PANview: A network monitoring system for the ”internet of things” http://ece.iisc.ernet.in/6panview, 2012.

[20] A. Paventhan, S.K. Allu, S. Barve, V. Gayathri and N. Mohan Ram, Soil Property Monitoring using 6LoWPAN-enabled Wireless Sensor Networks. In Third National Conference on Agro-Informatics and Precision Agriculture (AIPA), August 2012.

[21] S. Kuryla and J. Schönwälder, Evaluation of the Resource Requirements of SNMP Agents on Constrained Devices, In 5th International Conference on Autonomous Infrastructure, Management (AIMS), June 2011.

[22] H. Choi, N. Kim and H. Cha, 6LoWPAN-SNMP: Simple Network Management Protocol for 6LoWPAN, In 11th IEEE International Conference on High Performance Computing and Communications (HPCC), June 2009.

[23] Z. Shelby, Constrained RESTful Environments (CoRE) Link Format IETF RFC 6690, August 2012.