Smart Concrete:
Wireless Embedded Sensors to Monitor the Curing Temperature of Concrete

Jimmy Leahy
Kareem Al-Zooby
Guilherme Almeida do Nascimento

Illinois Institute of Technology

Embedded Systems and Reconfigurable Logic Design
Professor Jeremy Hajek
Dr. Daniel Tomal
April 30, 2014

Abstract

Information regarding the curing temperature of concrete is imperative in ensuring that it will reach its intended final strength. Failure to maintain acceptable curing temperatures will mean the concrete will not reach its intended final strength and can even render the concrete unacceptable, requiring the contractor to remove and replace it at their own expense. This information of concrete's curing temperature can also be used alongside a maturity curve to calculate the concrete's percent of final strength throughout the curing process.

For our project we created a mesh network of wireless temperature sensors that can be embedded directly into concrete at the time that that it is poured. These wireless sensors can transmit sensor data out of the concrete using radio signals where that information will then be passed down the network and displayed to the user. This will frequently provide the contractor with accurate information without requiring anyone to access hazardous or inconvenient locations. This information will also ensure that the concrete is curing within the acceptable range of temperatures and can provide the real-time maturity of the concrete, which will expedite the construction process by allowing the contractor to know exactly when forms can be stripped, loads can be placed, and the job can progress.

Introduction

Concrete is the most widely used substance in the world. Drastic improvements over the last 50 years to the strength, quality, variety, workability, and durability of concrete that we are able to produce has caused it to slowly but steadily dominate the industry in place of steel construction.

Concrete is mixed into a liquid and placed in it's liquid state. Once it is placed it will begin curing to its final strength. The curing process takes 28 days to complete, during which time close attention must be paid to maintaining the temperature and humidity of the concrete to ensure the final product will reach its intended final strength. Failure to maintain these properties will decrease its final strength and can render the concrete unacceptable, requiring the contractor to remove and replace it at their own expense.

During construction, concrete must mature to 70% of it's final strength before forms can be stripped and loads can be placed on it, allowing the job to progress. While concrete is widely used around the globe and has greatly improved in recent years, techniques for monitoring the humidity and temperature of it and for determining its maturity as it cures have not progressed far. Traditionally the method of ensuring that concrete cures within the acceptable range of temperatures, above 40F and below 125F, is by only pouring it on days that the outside temperature is acceptable.

The dominant method in use today for determining the maturity of concrete is by field curing test cylinders of the mix, then each day or so sending these cylinders off to a concrete testing company to be crushed and have their strength reported. Not only does this method require the employment of a testing company, but the number of results that can be provided are limited by the number of test cylinders and results are reported infrequently. These results are also rather inaccurate. Since the cylinders are cured outside the main pour in open air, they cure at a cooler temperatures and lower humidity than the pour they are supposed to represent.

Our Proposal

To accurately and frequently measure the temperature of concrete, a wirelessly device adapted to fit in small places is necessary. This device must also consume little power and be able to transmit data, even when surrounded by thick layers of concrete. The device presented in this project is used to measure temperatures (Figure 1) and incorporates the use of the XBee® ZB (http://www.digi.com/xbee) to capture data from the temperature sensors and send this data to the user. We chose to use the TMP36 for our temperature sensor and a single 3-volt coin battery to supply power to the circuit. All these components were soldered onto a small breadboard to ensure durability and minimize size.

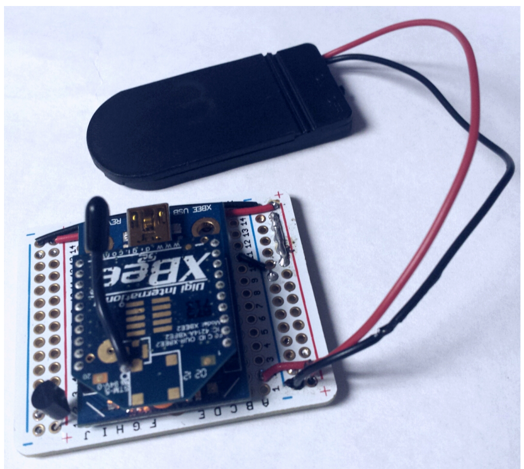


Figure 1

Main Components

The Xbee ZB2

XBee is a family of radio modules produced by Digi International [1]. In this project was used the XBee® ZB (Series 2) to transmit data from sensors to the central point. That version of Xbee allows the user to set up networks such as mesh networks, point-to-point, point-to-multipoint and peer-to-peer, besides of be “low-cost and low-power wireless sensor networks. The modules require minimal power and provide reliable delivery of data between remote devices” (Digi International, "XBee: Connect Devices To The Cloud").

Below are some of the main features of the XBee used in this project (Digi International, "XBee®/XBee-PRO® ZB RF Modules"):

- Indoor/Urban Range: Up to 300 ft.
- Outdoor RF line-of-sight Range: Up to 2 miles
- RF Data Rate: 250 kbps
- Supply Voltage: 3.0 - 3.4 V
- Operating Current: 45mA
- Operating Frequency: ISM 2.4 GHz
- Operating Temperature: -40°C to 85°C

XBee® ZB

As XBee modules meets all the requirements necessary for this project, such as being a low-power and low-cost device, we chose working with this product. Future versions may use other devices to transmit data, in case a less-cost or less-power device is found.

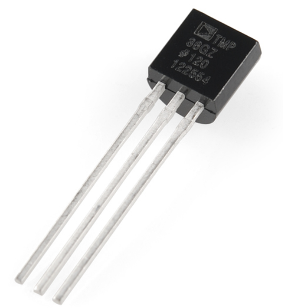
The Temperature Sensor

The sensor TMP36 from Analog Devices was used to measure the temperature of the concrete. This is a low voltage sensor which provides a voltage output that is linearly proportional to the celsius temperature scale. Its accuracy is ±1°C at +25°C and ±2°C over the −40°C to +125°C. This sensor fits perfectly in that project because of its low-cost and temperature range, once the concrete temperature varies from 4ºC to 51ºC.

Below are some of the main features of the TMP36 (Analog Devices, “Low Voltage Temperature Sensors - TMP35/TMP36/TMP37”):

- Supply voltage: 2.7V to 5.5V

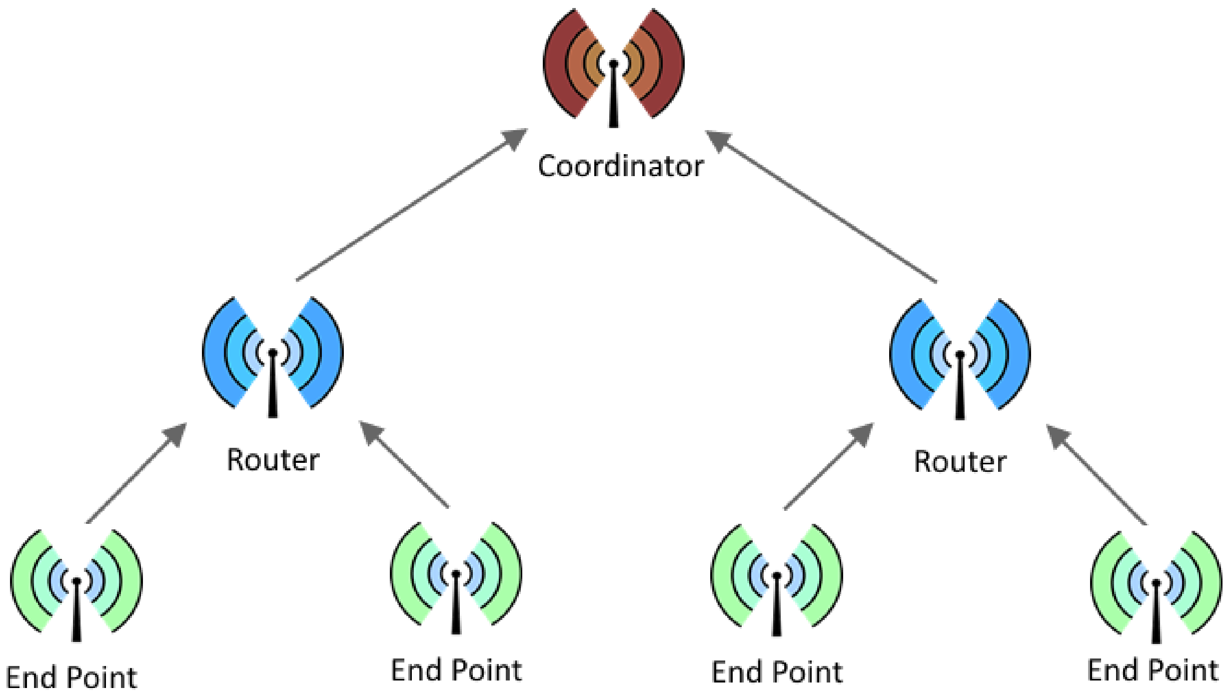
- Analog output (linearly proportional to the Celsius temperature)
- Calibrate directly in °C
- Accuracy: ±1°C at +25°C and ±2°C over the −40°C
- Operating Range: −40°C to +125°C



TMP36

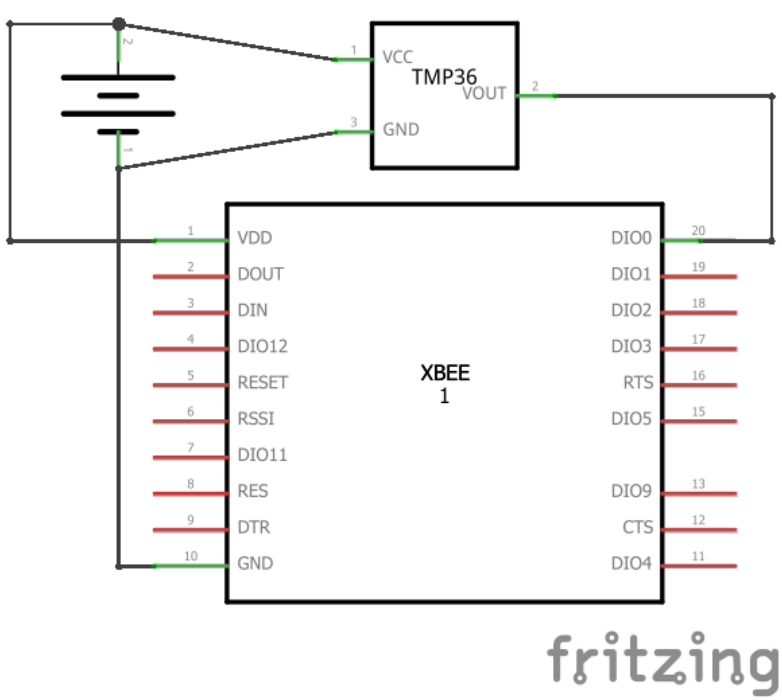
The Model Proposed

The model proposed to collect, transmit and receive data consists of a mesh network. A mesh network is a network topology in which each node relays data to other nodes in the network. Those nodes may work measuring a temperature value, repassing data, or simply receiving information. There are three types of nodes: end point, router and coordinator.



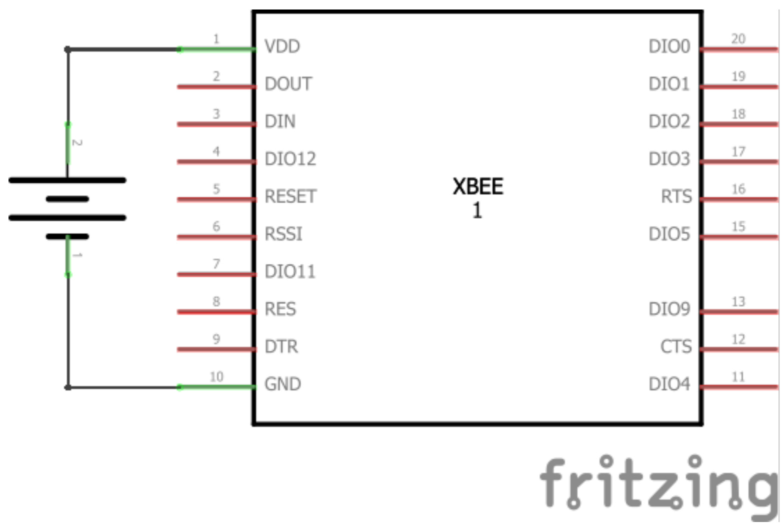
The End Point

The function of the end point is read the temperature sensor value and send it to the nearest router in the mesh network. Each endpoint has three main components: one temperature sensor (TMP36), one Xbee and one coin battery to supply the circuit. The diagram of the end point is shown below.



The Router

The router is composed of one Xbee and one battery. Its only function is to receive data from the end points and to send it to the coordinator. In our project we are using two routers, so that we can spread the endpoints in two different directions and the mesh network will keep working. Another advantage of using two routers is to hold the transmit of data in case one router stays out of order. In that case both routers should be close each other. The diagram of the router is shown below.



The Coordinator

The coordinator function is to receive data from the routers, pass it to an Arduino which function is decode the packets received and relays it to a computer which will send the main information (which endpoint is sending message and what is its local temperature) to a web server where the user can look at the information any time he wants. More details about the coordinator function are provided in the next pages.

Challenges

This project like most of other project faced a lot of kind of challenges and problem, which need to be handled. For example, choosing the right resources were a big challenge in order to save cost and energy, also founding the information and knowledge needed to implement chosen technology.

Energy and Cost Limitations

The nature of our project require long life devices and cheap construct cost. As result, we supposed to use technology that match this requirments. The bottom table shows some analysis that had been made by some people online on some networking technology which could be used in send and receive data. So as we can see the Zigbee technology, which is used by Xbee's, had the best attributes for our project requests. First, nodes per network xbee's compared to other technologies can handle a huge number of nodes up to 255/65k+ but Bluetooth and Wi-Fi can only 7 to 30 nodes per network.

Second the range, the Zigbee devices could send data in a range of 1-75+ meters. at the same time, the bluetooth is ranged between 1-10 meters and Wi-fi in range of 1-100 meters but it is expensive.

Third the battery life, one of the important challenges was keeping the lifetime long enough to complete the process. the Zigbee could stay for 100-1000+ days working using the sleeping mode option. on other hand, all other technologies cannot hold on more than week. also Xbee's is reliable, low power and cost, and effective.

	ZigBee™ 802.15.4	Bluetooth™ 802.15.1	Wi-Fi™ 802.11b	GPRS/GSM 1XRTT/CDMA
Application Focus	Monitoring & Control	Cable Replacement	Web, Video, Email	WAN, Voice/Data
System Resource	4KB-32KB	250KB+	1MB+	16MB+
Battery Life (days)	100-1000+	1-7	.1-5	1-7
Nodes Per Network	255/65K+	7	30	1,000
Bandwidth (kbps)	20-250	720	11,000+	64-128
Range (meters)	1-75+	1-10+	1-100	1,000+
Key Attributes	Reliable, Low Power, Cost Effective	Cost, Convenience	Speed, Flexibility	Reach, Quality

Xbee Configuration

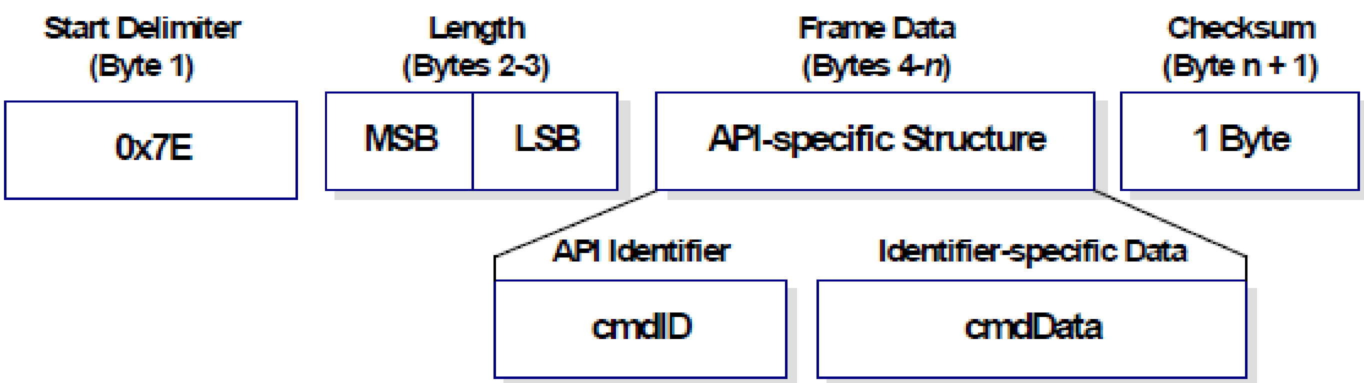
the Xbee is a pre programmable devices that just need to configure in order to control. There were a many option for configurations, but we had been using X-CTU for configuring the Xbee's, because it was easy, clear, and not using command line. So there were nothing to type just make selections. general speaking there is a three main models of Xbees with different types of firmware versions that can be used and updated.

Models: XBEE2, XBEEPRO2, PRO S2B.

In addition, we had been using this resources to configure and understand the nature of Xbees devices, also how it works.

- XBee S2 Quick Reference Guide/Cheat Sheet. (http://www.tunnelsup.com/xbee-s2-quick-reference-guide-cheat-sheet)
- Building Wireless Sensor Networks, by Robert Faludi.
- The manual describes of the XBee/XBee-PRO ZB RF module.

Moreover, after understand Xbee processing cycle, we needed to use one of the programing languages to process the messages that received from the XBee's. these packets had been divided into multiple frames, which have different types information about XBee's such as length, type, address, and payload.There are different ways to read the messages from XBee's. For example,processing or Arduino. We used the Arduino code to read and split the packets to a part and convert the payload data into temperature degree.the figure in the bottom show how the packets splitted first there is the identifier byte that always 0x7E, then there is two to three bytes for length. After there is the frame data which had two main parts cmdID and cmdData. First, the cmdID had the frame ID and type, and the source serial and network addresses. Second, cmdData this part carry the payload and and few information about the pins used to send data. Finally, there is the checksum for security purposes.



Code

```
void loop() {

if (Serial.available() >= 21) {
  if (Serial.read() == 0x7E) {

// read the variables that we're not using out of the buffer
    for (int i = 1; i<19; i++) {
// read (8,9,10,11) bytes to Identfy the address
      if(i == 8) { addr1 = Serial.read();

       }
        else if (i==9) { addr2 = Serial.read();

        }
        else if (i==10) { addr3 = Serial.read();

        }
        else if (i==11) { addr4 = Serial.read();

        }
 //     byte discard
        else { int discardByte = Serial.read(); }
    }
```

This sample code explain how did we read the packets to identify the address for each XBee. each XBee had two address, low and high. they were fixed address that cannot be changed. As result the XBee's used them as identifier by sending it encapsulated in the packets.

However, this sample show how to convert the two analog values, the low and high byte, to a voltage value that can represent values of temperature sensor.

```
float calculateXBeeVolt(int analogMSB, int analogLSB) {
  int analogReading = analogLSB + (analogMSB * 256); //Turn the two bytes into an integer value
  float volt = (((float)analogReading*1200) / 1023); //Convert the analog value to a voltage value
  return volt;
}
```

References

Low Voltage Temperature Sensors - TMP35/TMP36/TMP37. Norwood, MA: Analog Devices, n.d. PDF.

"XBee: Connect Devices To The Cloud - Digi International." *XBee: Connect Devices To The Cloud - Digi International*. Digi International, n.d. Web. 15 April 2014.

XBee®/XBee-PRO® ZB RF Modules. Minnetonka, MN: Digitally International Inc., n.d. PDF.