OTA Updates for an IOT Security Device

Aneesh Malhotra, Ryan Thomas, Sohail Iqbal, Gerson Dalton, Mohammad Nur

April 4, 2018

Motivation

- Wireless sensor network are becoming increasingly popular for both industrial and commercial use.
- FPGA's can be used to enhance security in wireless sensor networks by implementing a cryptographic exchange between wireless nodes in a network. [1] [2]
- Advancements in network attacks, however, could potentially leave these systems vulnerable.
- In the event the system is compromised it may be necessary to implement a firmware update.
- Over-the-air updates are those that occur wirelessly, and are easily distributed.

Current System

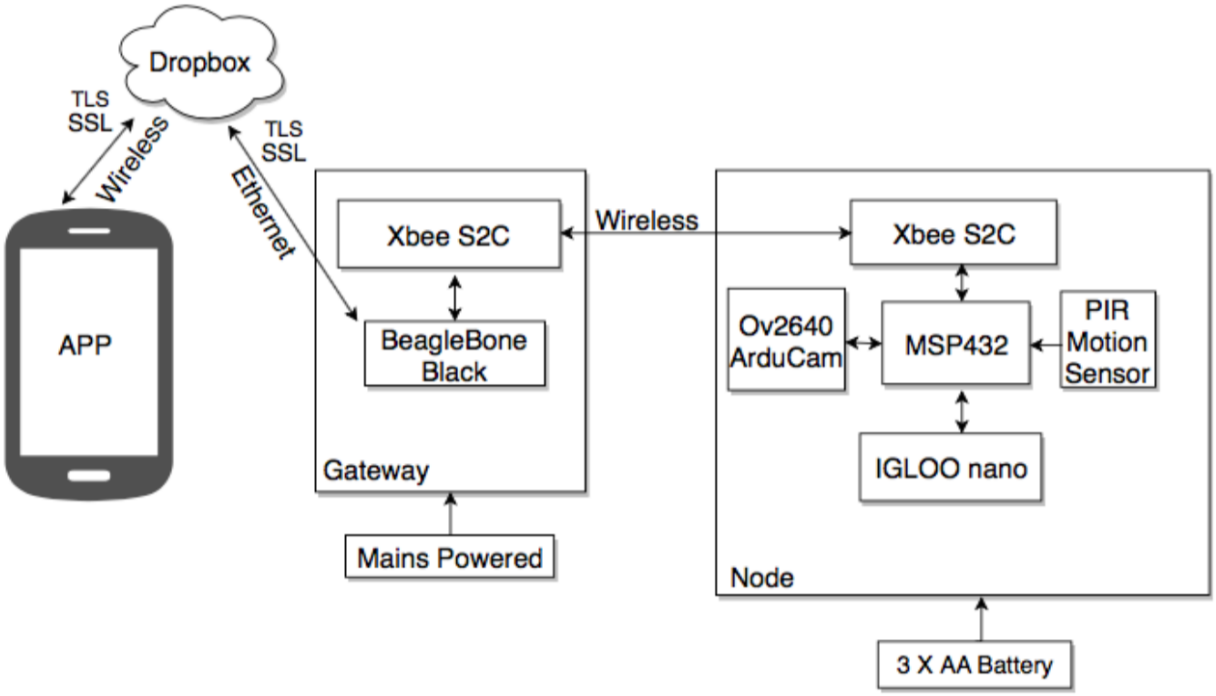


Figure: Top Level Design of IoT Security Device

Current System

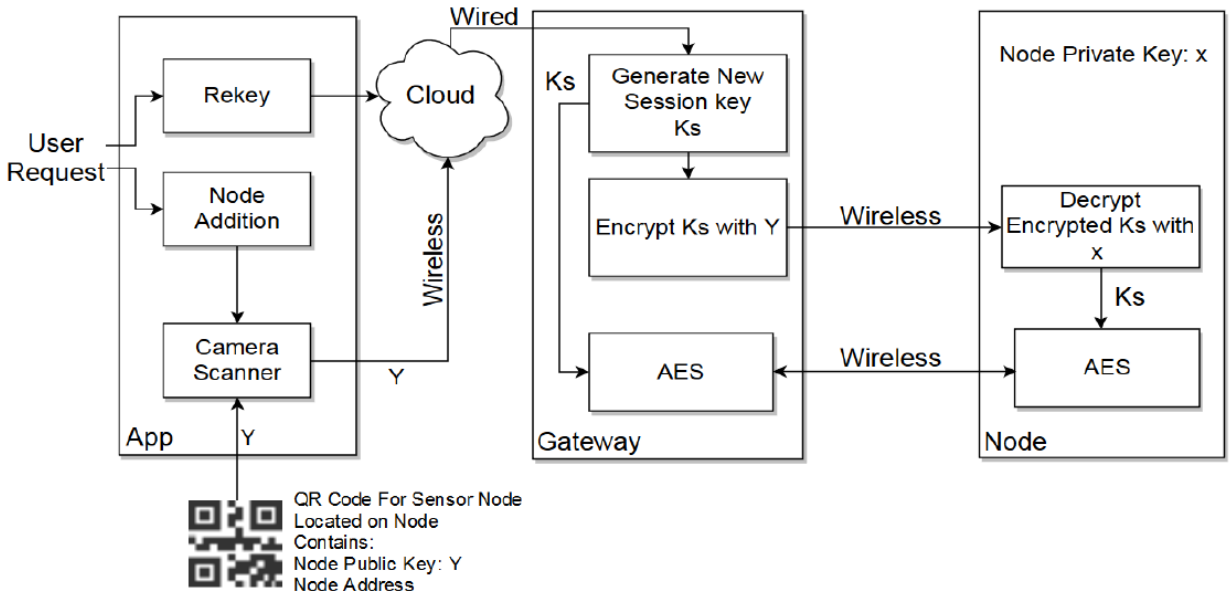


Figure: Functional Diagram of IoT Security Device

Problems

- This node consists of both a microcontroller and an FPGA, both of which need to be updated.
- We need to be able to distribute an update to the user.
- Gateway must be capable of receiving an update and pushing it to the node.
- The node must be capable of receiving an update from the Gateway and reprogramming itself.

Solution

Gateway

- Requests update from an external server, removing the need for Dropbox and increasing performance.
- Parses the update and sends bootloader commands to MSP432 corresponding to blocks of memory to be programmed

Node

- MSP432 will accept the update from the XBee
- The MSP432 will enter a bootloader and use instructions to update itself
- The MSP432 will interface to the FPGA via JTAG, and have a STAPL player for programming the FPGA.

System Design

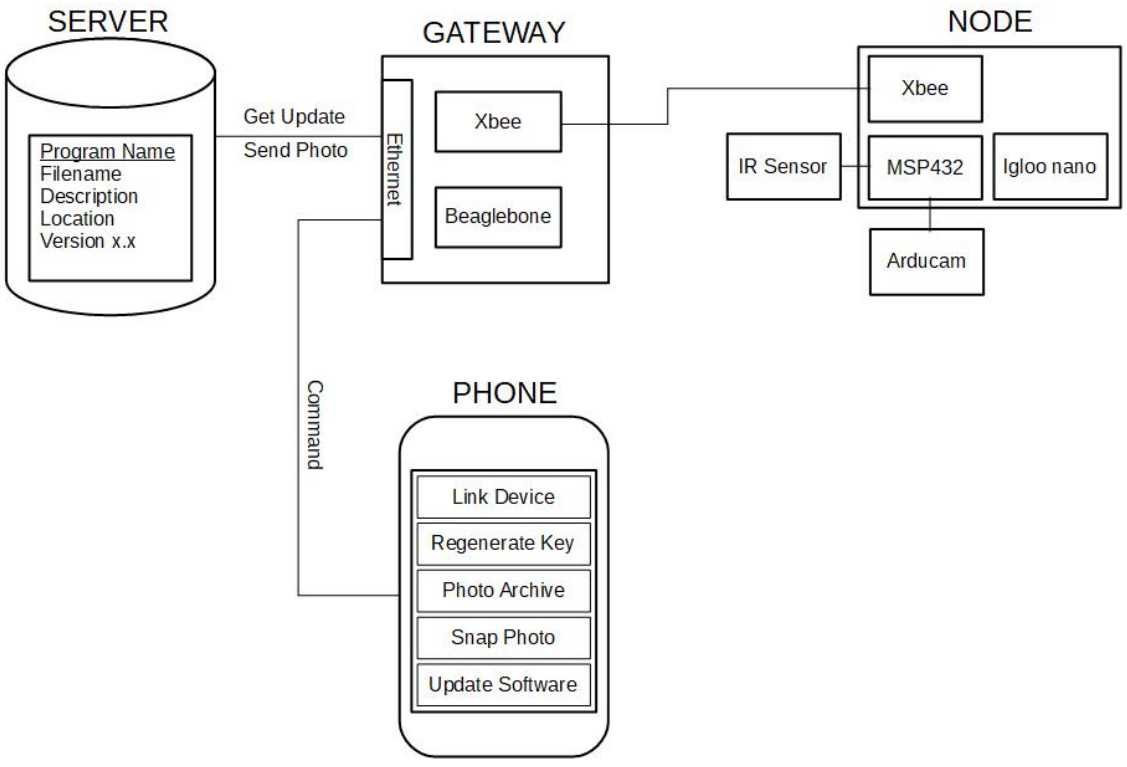


Figure: Top Level Design of OTA System with Server

Functional Diagram for Update

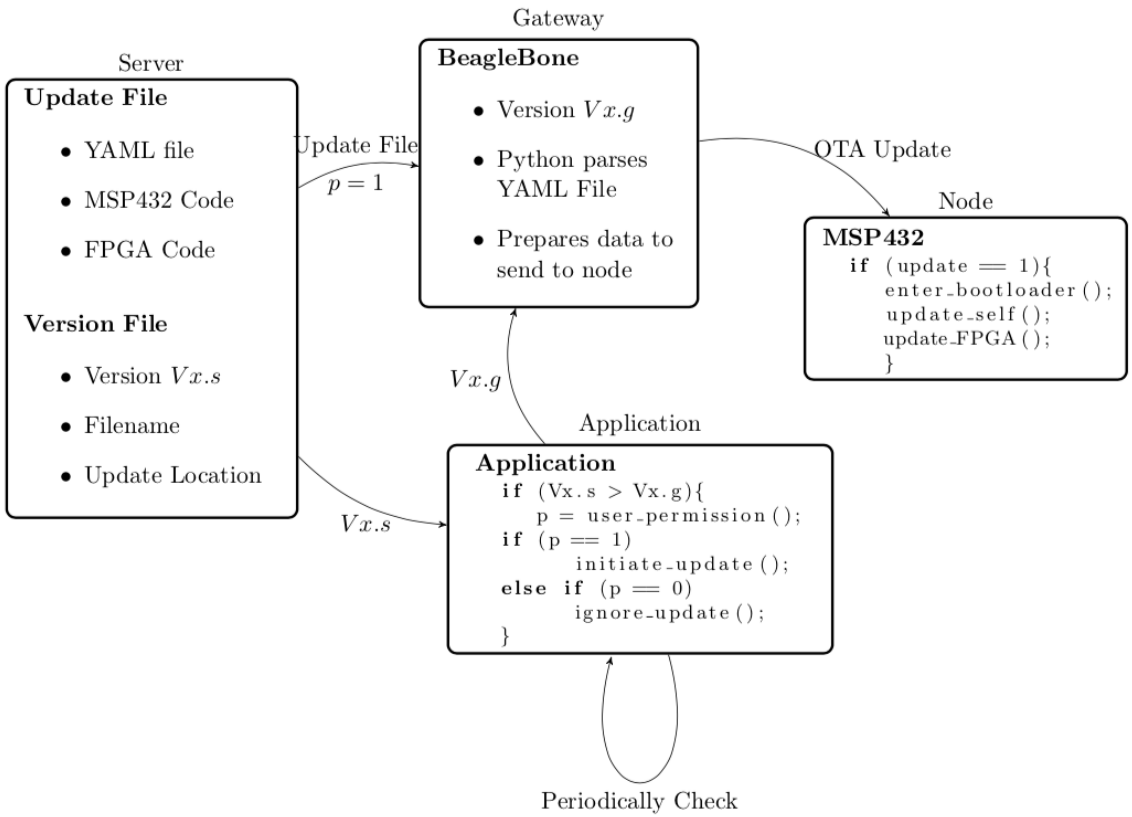


Figure: Functional Diagram of OTA Update Implementation

Application

Options:

1. Check for updates and request update permission
2. Request data from the node (image)
3. Re-Key option from previous project.

BeagleBone Black

1. Upon receiving update permission, the BeagleBone will download the update file from the external server.
2. The BeagleBone will parse the update information and send the update to the node via XBee.
3. The BeagleBone will also generate session keys as developed previously.

MSP432

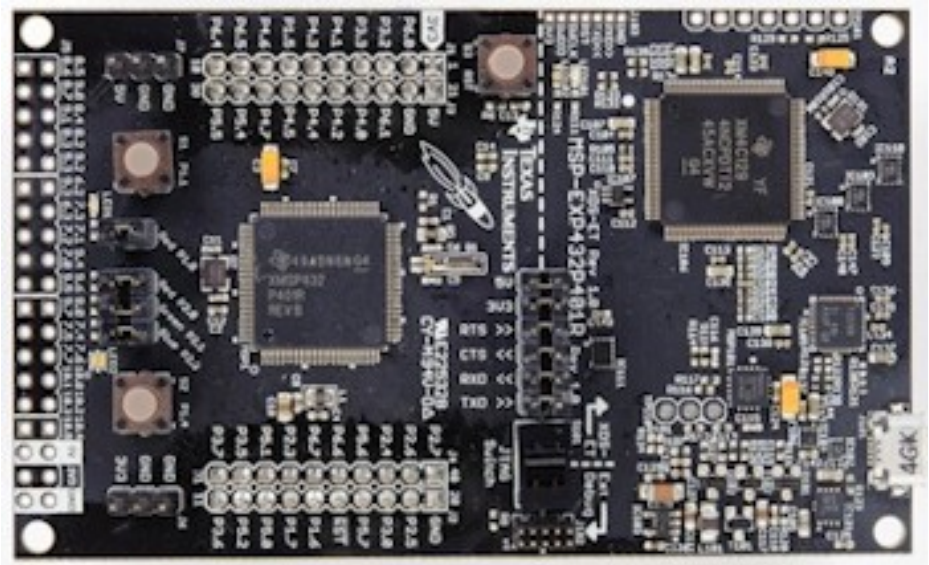


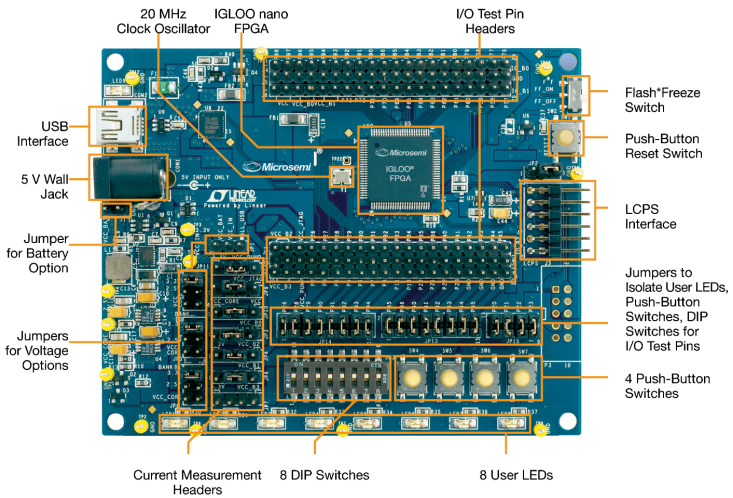
Figure: MSP432

MSP432

- The MSP432 will facilitate communication between all external peripherals on the node (XBee, IGLOO, ArduCam, IR Sensor).
- The MSP432 can be given a bootloader program which will allow us to access and write to program memory.
- The MSP432 can also be given STAPL commands to program the FPGA.
- We chose the MSP432 because the previous system required enough RAM to handle the transfer of images.

Actel IGLOO Nano

IGLOO nano Low Power Flash FPGA. Promises revolutionary and leading edge capabilities in less power consumption and smaller size. Also Allows in system programming using a micro controller (ISP).



Operating Modes

Sleep
ShutDown
Static Idle Mode
Flash freeze Mode
Active

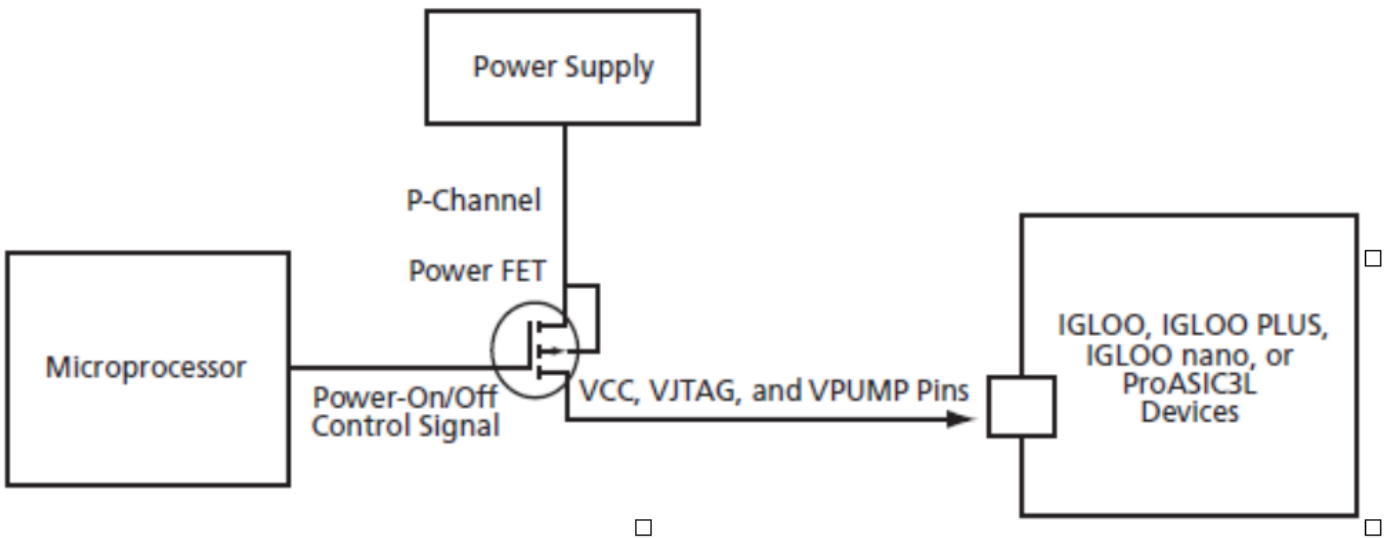


Figure: Operating Modes

Flash ROM

- The IGLOO nano contains 1kbit of *nonvolatile* Flash ROM memory.
- Flash Rom is arranged in 8 banks of 128 bits (128 Bytes).
- Each bank can be selectively programmed.
- Flash Rom can be read on Byte boundary

SRAM

- 8 Blocks each of 4,608-bit. (Total= 36864 bits or 4.608Bytes).
- Both the write width and read width for the RAM blocks can be specified independently for SRAM.
- The read and write clocks are completely independent.
- 2 synchronous read operations (pipelined and non-pipelined) and 1 write operation for SRAM.

FGPA Programming

Two ways to program: Device Programming (not used)
With Internal System Programming (ISP) (our method)

- JTAG interface on FPGA through JTAG pins. Pins can be controlled by either processor or programmed through a header connection.
- Both the write width and read width for the RAM blocks can be specified independently for SRAM.
- It can be programmed repeatedly and erased even if it is mounted. If the application requires it, the system can be designed to reprogram itself using a microprocessor, without the use of any external programmer.

How to Program

- First download the C-based STAPL player or DirectC code from the website.
- Create the low-level application programming interface (API) to provide the necessary basic functions.
- These API functions act as the interface between the programming software and the actual hardware.
- The API is then linked with the STAPL player or DirectC and compiled using the microprocessor's
- Compiler. (We wont use DirectC since it does not support the upgrading).
- After compiling download the resulting binary into the MCU system's program memory (such as ROM, EEPROM, or flash).
- Download Bitstream or STAPL file created using the Microsemi Designer
- software(Libero) into the MCU system's volatile memory.
- Activate the stored programming binary file.

Update

Updates will consist of:

1. A YAML file with instructions for the bootloader
2. A section containing the MSP432 update
3. A section containing the IGLOO udpate
4. An update will be initiated by the App

YAML

- YAML is a serialization language that will allow us to map the update to appropriate memory blocks
- This will allow us to update a specific block of memory at a time.
- The processing of the YAML file will be done on the BeagleBone.

Encryption/Security

- OTA update capability can provide a potential vulnerability to the system.
- We need a system of authenticating updates as to prevent the system using a false update.
- We can do this using a digital signature scheme through RSA or ECC.

Memory

- The MSP432 has 64kB RAM.
- Both the MSP432 and FPGA have 256kB of flash memory.
- The size of the current system is 222kB, and the picture size is a maximum of 60kB
- In order to be able able to support an update we need to have an external memory pool.
- The size of MSP432 firmware is 7.5kB

Data Protocols

- Our wireless sensor node will use the MSP432, which is a low power microcontroller for processing and redirecting sensor data.
- The FPGA can be made handle the same data protocols as the MSP432 such as I^2C, SPI, and UART.
- I^2C and UART protocols require 2 pins, while SPI requires 4.
- I^2C and SPI are synchronous transfers, whereas UART is asynchronous and does not require a clock.
- SPI can transfer at 16Mbps
- UART can transfer at 960kBps

Zigbee

- A high level communication protocol based in IEEE 802.15.4, which is used in low rate wireless personal networks.
- ZigBee is common in IoT devices as an alternative to WiFi.
- This protocol will be implemented using the XBee devices on the Gateway and Node, and will facilitate data transfer between them.

Pros of Zigbee

- It is designed for small devices for low power consumption.
- It is a mesh network standard which can use other devices to pass signals over long distances.
- The data rates vary from 20kbps – 250kbps in the 2.4GHz band.
- Has good performance in environments with low SNR.
- Secures data through 128-bit symmetric encryption keys.
- Needs less than 64kb of ROM and 2 – 32kb of RAM.

PCB Design

For the PCB Design we will need to take into consideration the following:

- Use KiCad to develop a circuit schematic for the PCB.
- Redesign the PCB to allow for easier measurement of system specifications such as power consumption.
- This will require us to fix the previous errors.

Next Steps

We will need to test each component individually

1. Getting device to enter bootloader by reading an update sequence
2. Tell the bootloader to write to a specific memory block
3. Begin developing

Tests to Run

1. We will test our final results by creating an update file, which will call for a noticeable update for both the FPGA and MSP432
2. This update could involve changing the state of an LED.

Division of Labor

Aneesh

- Project manager responsibilities such as keeping everyone on schedule.
- Create update file, and update verification
- Work on understanding YAML

Ryan

- Mainly responsible for writing MSP432 code to be able to reprogram itself
- Working with the MSP432 bootloader.

Division of Labor

Gerson

- Create PCB design of final system, for both the gateway and the Node.
- Handle any issues interfacing via ZigBee to any of the devices

Mohamed

- Working on the application, which will interface to the server, to communicate with the gateway.
- Helping write code to update the BeagleBone on the Gateway

Sohail

- Working on downloading and implementing STAPL player
- Writing API to be able to interface to the FPGA via JTAG.

References

J. Sen, "Security in Wireless Sensor Networks," *arXiv:1301.5065 [cs]*, Jan. 2013, arXiv: 1301.5065. [Online]. Available: http://arxiv.org/abs/1301.5065

L. G, S. K, and V. K. R, "Elliptic Curve Cryptography implementation on FPGA using Montgomery multiplication for equal key and data size over GF(2m) for Wireless Sensor Networks," in *2016 IEEE Region 10 Conference (TENCON)*, Nov. 2016, pp. 468–471.