Alfonso Meraz, Alexander Melnick
EC535 - Embedded Systems
Prof. Babak Kia Montazam
Technical Report

Introduction

Our project aimed to develop a wirelessly controlled locking mechanism, utilizing NFC tokens as the keys for unlocking. To achieve this, we conducted an in-depth study of various Near Field Communication (NFC) controllers to select an appropriate peripheral that would align with the constraints of our communication interface and microcontroller board. The system comprised a BeagleBone Black microcomputer and NFC tokens, with a solenoid serving as the locking mechanism. This setup allowed us to explore the practical applications and integration of NFC technology in security systems.

Project

The primary objective of our project was to explore and understand the communication interfaces between peripheral devices and our BeagleBone microcontroller. We aimed to demonstrate this through the application of a locking mechanism. Initially, we outlined all software requirements and proceeded to design the hardware layout. Our system required the development of several kernel modules, including a main kernel module that integrated the communication interface, solenoid, and NFC token tag drivers. We crafted specific modules for the NFC tag, solenoid, and multiple communication interfaces. Challenges with the BeagleBone board and the NFC controller led us to develop both SPI and I2C interface drivers.

Prior to assembling the hardware, we familiarized ourselves with the necessary circuitry through YouTube videos and online tutorials. The design for the solenoid and NFC controller included pull-up resistors, controlled voltage sources, and transistors for kernel management. The circuit for the solenoid also incorporated a diode and transistor to regulate the voltage and current. We experimented with two different NFC controllers; one supported only SPI while the other was compatible with SPI, I2C, and UART. Initial attempts to develop an SPI driver were thwarted by compatibility issues with our BeagleBone’s kernel image, prompting us to switch to an I2C interface, necessitating the acquisition of an alternative NFC controller board and the development of an additional kernel module for I2C.

Throughout the project, we engaged deeply with SPI and I2C protocols as well as various specification sheets. While we did not fully realize our original vision, the project was invaluable in teaching us about the critical elements of embedded systems, from design and technical specifications to selecting appropriate hardware. This experience has significantly enhanced our understanding and skills in the field of embedded system design.

Challenges

Our project faced numerous challenges from the outset. Initially, we aimed to establish communication between our BeagleBone and the MFRC522 NFC Reader/Writer using SPI. Although we successfully implemented SPI communications and the Self-Test procedure outlined in the MFRC522 documentation, we discovered that our kernel lacked the necessary SPI drivers. We explored several remedies: first, we recompiled our kernel based on procedures from Lab 1. Although successful, this kernel was too basic for operational use.

We then considered using the standard Debian distribution from the BeagleBoard website, which introduced two new issues: the inability to transfer files to the board and the inability to compile our modules after transfer. The distribution lacked a serial interface and was incompatible with our cross-compiler. We resolved the file transfer issue using a USB drive, but the absence of compiler support remained unsolved. Attempts to compile directly on the board were only successful for userspace programs, not kernel modules.

Confronted with these SPI difficulties, we shifted our focus to interfacing the MFRC522 via I2C. This option was less desirable as it required discarding our existing SPI code. While we prepared a preliminary module to test the I2C connection, we discovered that our board's MFRC522 was specifically hardwired for SPI. Online research suggested that modifying the hardware by cutting a specific pin could switch the connection mode, but given the tiny size of the IC (approximately 1 square centimeter), we opted to switch to another board rather than risk irreversible damage.

Our subsequent switch to the PN532 board introduced even more challenges. The PN532's memory map was exceptionally complex, involving both 8-bit and 16-bit I2C addresses. Despite exhaustive efforts, we encountered inconsistent and puzzling results. We tried debugging using built-in Linux functions like i2cget, i2cset, and i2cdetect, yet we couldn't detect the board with i2cdetect and only read zeros from the registers with i2cget—values that offered no meaningful information. As a final measure, we connected the PN532 to an Arduino, using code sourced online that was purported to work. This test confirmed our fears: the board was only partially functional, unable to detect or read any NFC tags.

Discussion

Successes

We achieved notable success in our project by effectively deploying a kernel module to control our solenoid-based locking mechanism. Building upon the foundations laid in Lab 4, we were able to successfully activate and deactivate our "lock." This achievement necessitated the acquisition of new knowledge in electronics, an area previously unfamiliar to both of us. Specifically, we delved into the workings of solenoids, flyback diodes, and discrete transistors. These concepts were crucial in enabling us to understand and manage the physical components of our locking system effectively.

Failures

Despite our efforts, we were unable to successfully read or write to our NFC boards. The challenges posed by various interfaces were insurmountable within the constraints of our limited time and resources. Our frequent switches between different boards, interfaces, and kernel configurations ultimately prevented us from achieving the level of interaction needed to communicate effectively with the NFC tags. Consequently, our locking mechanism remained inoperable, as it lacked any keys or a user-accessible mechanism to control it. This aspect of the project underscored the complexities of integrating hardware and software in embedded systems and highlighted the critical importance of stability and consistency in technological configurations.

What we Learned

Throughout this project, we acquired numerous valuable skills. Our investigation into the SPI and I2C communication protocols included an in-depth study of their specifications, as well as extensive engagement with the Linux/I2C and Linux/SPI libraries and their various implementations. The datasheets for both the MFRC522 and PN532 were meticulously analyzed, enhancing our understanding of these devices through repeated readings.

Additionally, we gained practical, hands-on experience. For instance, we learned to solder when we needed to attach headers to the MFRC522 board—a skill that was new to both of us. Our knowledge of electronics also broadened significantly, particularly through our work with a solenoid. This introduced us to the practical application of a flyback diode, deepening our understanding of electronic circuit design and component function. These experiences have enriched our technical proficiency and prepared us for more advanced projects in embedded systems.

Conclusion

In conclusion, reflecting on our project, there are several aspects we could have approached differently from the start, potentially leading to more success in meeting our original objectives. Key among these was the planning and selection of hardware devices and communication interfaces. A deeper initial understanding of these interfaces and a thorough evaluation of their respective advantages and disadvantages might have guided us to choose the most suitable Near Field Communication controller for our needs.

Despite these challenges, the project proved to be a valuable learning experience. We gained significant knowledge from datasheets, learning how to configure devices within specific constraints and design parameters. We also expanded our technical skills by constructing circuits that were initially unfamiliar to us, such as those involving the solenoid and NFC controller, as well as managing kernel images for our BeagleBone device.

This project has undeniably enhanced our ability to tackle new challenges in embedded systems development. With the insights and skills we have acquired, we are now better equipped to embark on future projects with greater confidence and expertise.

References

1. Corbet, J., Rubini, A., & Kroah-Hartman, G. (n.d.). SPI - Serial Peripheral Interface. Retrieved from https://www.kernel.org/doc/html/v4.9/driver-api/spi.html
2. LinuxTV. (n.d.). API struct spi_board_info. Retrieved from https://linuxtv.org/downloads/v4l-dvb-internals/device-drivers/API-struct-spi-board-info.html
3. Embetronicx. (n.d.). Solenoid wiring tutorial [Video]. YouTube. https://youtu.be/Yh3TLXihUps?si=wRjqsa_WysqMT6_W
4. Embetronicx. (n.d.). Linux Kernel SPI Device Driver Tutorial. Retrieved from https://embetronicx.com/tutorials/linux/device-drivers/linux-kernel-spi-device-driver-tutorial/
5. Embetronicx. (n.d.). SPI - Serial Peripheral Interface Protocol Basics. Retrieved from https://embetronicx.com/tutorials/tech_devices/spi-serial-peripheral-interface-protocol-basics/
6. Embetronicx. (n.d.). I2C Tutorial Part 1. Retrieved from https://embetronicx.com/tutorials/tech_devices/i2c_1/
7. Embetronicx. (n.d.). I2C Tutorial Part 2. Retrieved from https://embetronicx.com/tutorials/tech_devices/i2c_2/
8. Embetronicx. (n.d.). I2C Linux Device Driver using Raspberry Pi. Retrieved from https://embetronicx.com/tutorials/linux/device-drivers/i2c-linux-device-driver-using-raspberry-pi/
9. Embetronicx. (n.d.). I2C Bus Driver Dummy Linux Device Driver using Raspberry Pi. Retrieved from https://embetronicx.com/tutorials/linux/device-drivers/i2c-bus-driver-dummy-linux-device-driver-using-raspberry-pi/
10. Linux Kernel Documentation. (n.d.). Writing I2C Clients. Retrieved from https://docs.kernel.org/i2c/writing-clients.html
11. UnboxnBeyond. (2020, June 12). I2C Communication in BeagleBoneBlack. Retrieved from https://unboxnbeyond.wordpress.com/2020/06/12/i2c-communication-in-beagleboneblack/
12. Onion Corporation. (n.d.). Digging into i2cget and i2cset. Retrieved from https://onion.io/2bt-digging-into-i2cget-and-i2cset/