ADXL345 Knock Pattern Sensor

Allison Pearce
St. Edmund's College
ap819@cam.ac.uk

I. Introduction

This paper describes a low-power knock pattern sensor built using an Atmel ATmega 644P microcontroller [1] and an ADXL345 triple axis accelerometer [2]. Knock pattern sensors have a number of useful applications. Shock sensors are often used to detect forcible entry into roadside telecommunications cabinets, and when a knock sensor is present, authorized personnel can enter a tap code to override the alarm. Knock patterns can also be used as an unlocking mechanism for a door or safe. Tap authentication has been investigated as a replacement for passwords in mobile devices, indicating that tap authentication is comparable to industry standards for security and usability with the added advantage of being inconspicuous [3]. Tapping wireless devices together can also form part of a protocol for communication and information transfer [4]. For example, users can tap two phones together to share an image from one to the other.

The goal of this project was to make a functional knock pattern sensor that could compare a reasonably complex knock pattern to a previously recorded pattern. This required sensitive tap detection with minimal false positives and a method of representing and comparing knock patterns. The device also needed to be low power and user friendly.

II. Preparation

A. Device Selection

This project required a microcontroller and a sensor to perform knock detection. I selected the Ateml ATmega 644P for the microcontroller because it is easy to use and supports the SPI communication needed for the accelerometer. It has several features for reducing power use. The system's storage requirements were low, and the ATmega 644P's memory was more than sufficient. I was also familiar with it from previous P31 practical work.

The ADXL345 triple axis accelerometer was chosen as the knock sensor for its sensitivity and ease of use. Alternatives included shock sensors and other types of accelerometers. An accelerometer was preferred to a shock sensor because previous implementations of knock pattern sensors reported that shock sensors were difficult to interpret. Compared to other accelerometers, the ADXL345 provides several unique and advantageous features. Chief among these is its built-in tap detection function. Taps are defined by several parameters that can be used to tune the sensitivity of the knock sensor, and detected taps can trigger an interrupt in the microcontroller. The ADXL345 is a digital accelerometer that uses SPI or I^2C to communicate with the microcontroller. It has one of the lowest accelerometer power requirements. The ADXL345 also provides programmable power modes than can be used to further reduce power consumption.

III. Implementation

A. System Overview

The device has two modes of operation, one for recording passcodes and one for comparing input knock patterns to the stored pattern. The device is initialized in recording mode. The user knocks the pattern, and a green LED flashes as each knock is registered. After three seconds of inactivity, the recording period times out, the red LED flashes the pattern back to the user for verification, and the device enters listening mode. In listening mode, the device sleeps until interrupted by a knock. It will record any subsequent knocks until three seconds pass without activity. Again, the green LED flashes with each tap. It then compares the new pattern to the previously recorded pattern. A green LED flashes if the pattern is correct, otherwise a red LED flashes. To record a new pattern, the user presses the button switch until the LEDs flash in a red-green-red pattern.

B. Hardware

In addition to the microcontroller and accelerometer, this project required a 5V 100mA power supply, a prototyping board, wires, a programming cable, a four-panel switch, two LEDs, a 1K Ohm resistor, and a button switch. The resistor was used to pull down the button. The four-panel switch was necessary because the accelerometer communicates with the microcontroller over SPI, which uses the same pins (MOSI, MISO, SCK) as the programmer. The microcontroller, accelerometer, and other components were connected as shown in Figure 1.

C. Software

The C program executed by the microcontroller consists of several initialization functions, a simple main loop, interrupt handlers, functions to record and compare knock patterns, and functions for communicating with the user via LED flashes. Some of the functions for communicating with the ADXL345 were taken from an example provided on the P31 website.

The initialization functions establish the communication method between the ATmega 644P and the accelerometer, set values for the tap detection parameters and other ADXL345 registers, set parameters for the timer, and enable interrupts. The two most important parameters for tap detection are the tap threshold and the window. The tap threshold determines the minimum acceleration value that is considered a tap, and the window is the maximum time that the acceleration can remain above the threshold. Adjusting these parameters makes the system more or less sensitive to knocks. The other ADXL345 registers that were set and their values are described in Table I.

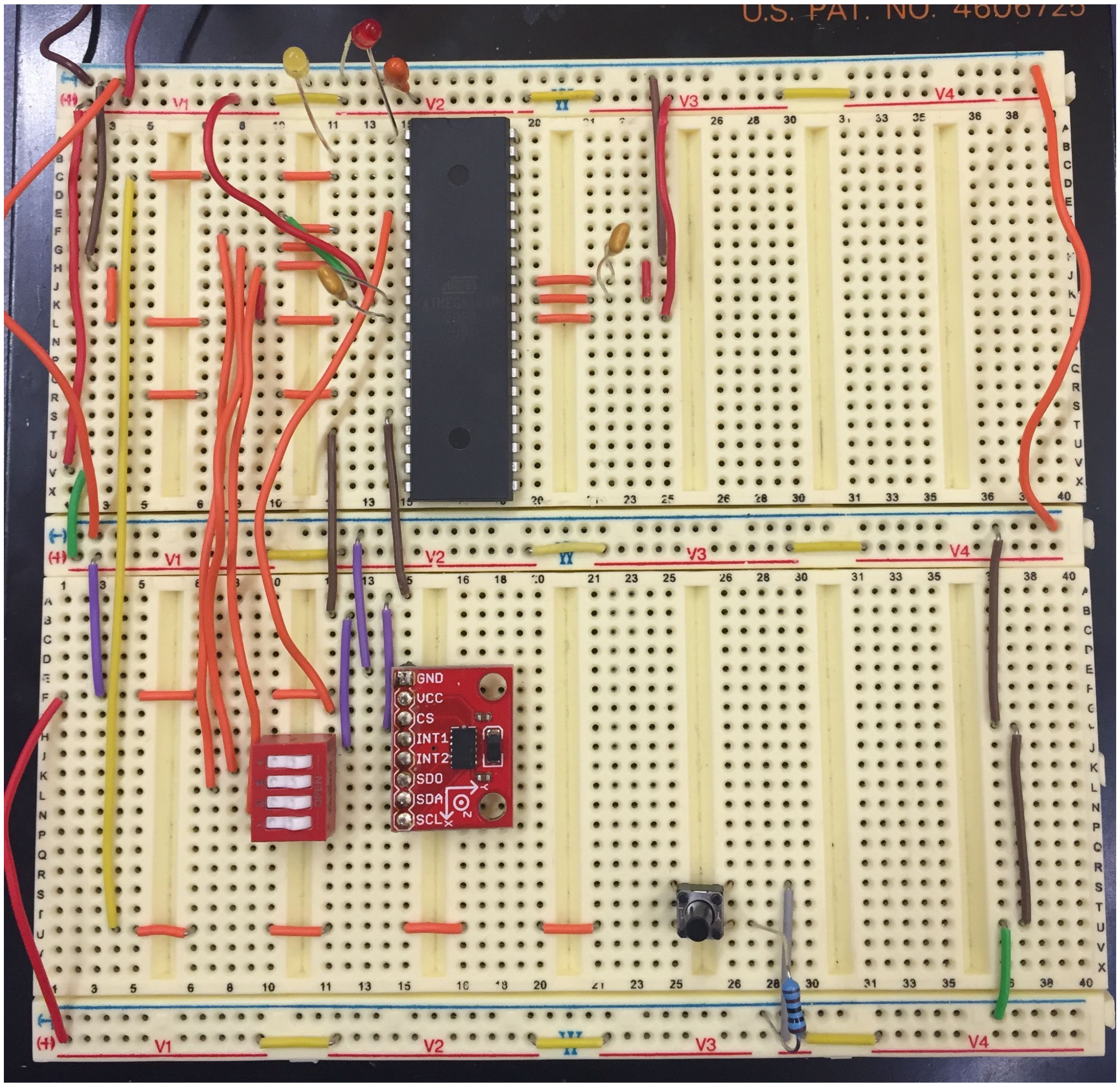

Fig. 1. Connection of components on prototyping board

TABLE I. ADXL345 REGISTERS

Register	Description	Value
`THRESH_TAP`	Minimum acceleration value for taps. Scale factor 62.5mg/LSB	70
`DUR`	Maximum time event can be above threshold to be considered a tap. Scale factor 625μs/LSB	16
`LATENT`	For double tap detection. Set to 0 to disable double taps.	0
`WINDOW`	For double tap detection. Set to 0 to disable double taps.	0
`TAP_AXES`	Which axes to use in tap detection.	0x07
`INT_ENABLE`	Which interrupts to enable	0x40
`INT_MAP`	Map interrupts to microcontroller pins	0x40

The bulk of the work in the software is triggered from the tap interrupt. Upon receiving the first tap after a period of inactivity, the tap interrupt starts the ATmega 644P's built-in 16-bit timer. Subsequent tap interrupts cause the timer value to be recorded and then clear and restart the timer. A timer interrupt is used to prevent overflow errors in the timer's counter. The timer interrupt is also responsible for calling the appropriate function after three seconds of inactivity: in record mode, it flashes the pattern back to the user, and in verify mode, it compares the input pattern to the recorded pattern and flashes the green or red LED depending on if it was correct.

Knock patterns are defined by the intervals between taps and stored as arrays of integers. When comparing patterns, the new pattern is immediately invalidated if it has a different number of knocks than the code pattern. The new pattern is accepted if it has the same number of taps as the code and each interval is within an acceptable range of the code's interval for that position in the sequence. The range can be changed to tune the sensitivity of the device, but 20ms was determined to be a reasonable default.

The main loop checks to see if a button press has occurred, in which case the mode (record or verify) is toggled. If not, it sets the sleep mode and puts the device to sleep. The full program is included in the Appendix.

One optional feature of the ADXL345 that was disabled in software is double tap detection. It is possible to make a distinction between single taps and double taps by defining additional parameters and flipping bits in the appropriate registers. Handling single and double taps differently offered no significant advantages for distinguishing different knock patterns, but it would have increased the complexity of how knock patterns are stored and compared.

D. *Power Concerns*

Low power consumption was a primary goal of the system achieved through a combination of power reduction strategies. The microcontroller enters ADC noise reduction sleep mode whenever three seconds pass without a knock interrupt. This sleep mode provided the greatest possible power reduction while still allowing the microcontroller to respond to knock interrupts. The accelerometer operates in a power saving mode, with all unnecessary functions turned off in the Power Reduction Register (PRR). TWI, Timer/Counter2, Timer/Counter0, USART1, and USART0 were disabled. Timer/Counter1 and SPI remained on because they were necessary for knock timing and communication with the accelerometer. Another simple strategy that yielded a huge power reduction was adding pull-up resistors on all inputs to prevent them from floating. Though the ADXL345 offers a sleep mode, it was not used because it interfered with tap detection and triggering interrupts in the microcontroller.

E. *Challenges*

A difficult step early in the development process was configuring tap detection on the ADXL345. Most of the process is easy to follow, but I did not initially realize that the `INT_SOURCE` register had to be read in the main loop in order for the tap interrupt to fire. Configuring all of the values for the different tap detection registers also required a reasonable amount of experimentation.

Another step that was less straightforward than expected was determining the proper sleep mode. Based on the documentation for the ATmega 644P and the ADXL345, I expected all of the 644P's sleep modes to work because the INT1 interrupt should be able to wake the microcontroller in all of them. In practice, however, I found that tap interrupts did not wake the microcontroller unless I used ADC noise reduction sleep mode.

Finally, the use of the four-panel switch to allow the accelerometer and programmer to communicate was a technique I had not encountered before, but thanks to the teaching staff, that problem was solved quickly.

IV. Evaluation

The goal of the project was to make a low-power knock pattern sensor that was accurate and easy to use. All aspects of this goal were accomplished.

A. *Power Consumption*

Before any measures were taken to reduce power consumption, the device drew between 0.017mA when idle and 0.031mA when flashing the LEDs. The combination of all power saving strategies reduced these values to 0.004mA and 0.019 mA, respectively. The power savings achieved for each method in isolation are shown in Figure 2.

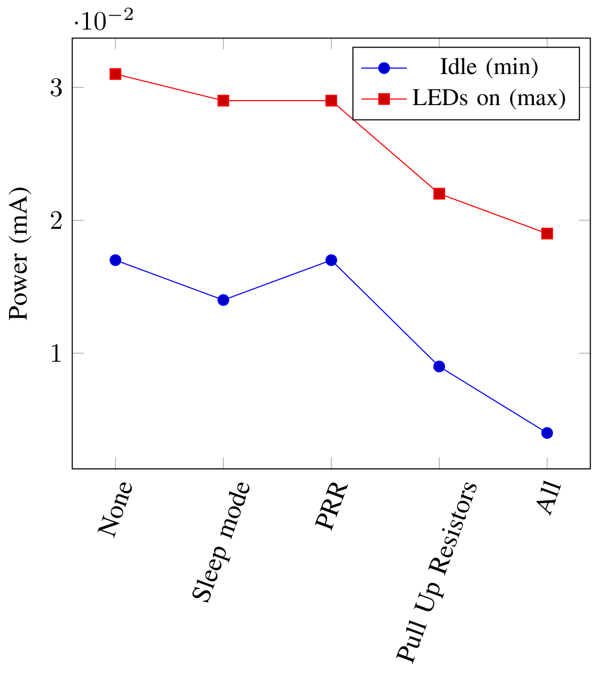

Fig. 2. Power use for different power saving techniques. The red line represents maximum current, when the LEDs were on, and the blue line is minimum current use, when idling between taps.

B. *Accuracy*

The accuracy of the device can be affected to some degree by manipulating the sensitivity of the knock sensor. This is achieved by changing the tap threshold, tap window, and the size of the window used to compare knock intervals. A less sensitive knock sensor would register fewer false positive knocks and might perform better in a noisy environment, but it would require the user to be more deliberate about knocking, because lighter taps would not register. This might make it more challenging to input certain types of knock patterns. A more sensitive sensor, on the other hand, might interpret an accidental hand slip as a knock. Therefore it is better for accuracy if the user is aware of the device's settings and how they affect the expectations for knocking.

Using the default settings from Table I, the device was tested extensively and there were no false positives or false negatives due to the device's error. Human inability to tap in the same pattern multiple times, especially after tapping several intentionally incorrect patterns, was the primary source of error in testing. I also experimented with one user recording the passcode and another user attempting to enter it. Again, the device performed perfectly when the humans involved did not make mistakes.

C. *Ease of Use*

The device was not intended to be a complex system, and it provides all the functionality and feedback required with just one button, two LEDs, and the knock sensor. Resetting the code is a simple process, requiring only a button press and demonstration of the new pattern. The meanings of the various LED flashes are intuitive (green flashes to register knocks, red to replay the pattern, green or red flashes when reporting whether the pattern was correct).

V. Future Work

The sensor described in this paper provides all the functionality needed to record and detect knock patterns, but several more sophisticated features could be added. One useful extension would be the ability to store multiple patterns in different categories. This would be useful if the sensor was used to open a door, for example. The owner of the building could have a master pattern and the option to add a one-time passcode for a contractor or a passcode that expires after a set time to be used for guests coming to a party. Other extensions of password modules could also apply, such as providing hints if the user forgets the code. A more sophisticated user interface could be created by adding an LCD screen and a keypad for user inputs other than knock patterns. However, these enhancements would significantly increase the memory and power requirements. Future work could also include incorporating the sensor with an application—attaching it to a servo-based locking mechanism and using it to unlock a door, or integrating it with a laptop or phone to replace the login password.

VI. Summary

I implemented a low-power knock pattern sensor that is simple to use and adaptable to a variety of applications. The system was built using an Amtega 644P microcontroller and an ADXL345 triple axis accelerometer. The device is highly accurate and the level of sensitivity can be customized to increase security or to reduce interference from noisy environments. The ADXL345 was an excellent choice for the knock sensor, especially because it provides an easily configured tap detection function. The code for storing and comparing knock patterns was both simple and effective, and did not seem to limit the complexity of knock patterns that could be recorded and verified. This project helped me learn more about microcontrollers, circuits, and their debugging strategies.

References

[1] San Jose Atmel Corporation. ATMega644p Datasheet. *CA, October*, 2006.

[2] Analog Devices. ADXL345 Accelerometer Datasheet, 2011.

[3] Diogo Marques, Tiago Guerreiro, Luís Duarte, and Luís Carriço. Under the table: Tap authentication for smartphones. In *Proceedings of the 27th International BCS Human Computer Interaction Conference*, BCS-HCI '13, pages 33:1–33:6, Swinton, UK, 2013. British Computer Society.

[4] O.K. THORN. Interacting with devices based on physical device-to-device contact, July 9 2013. US Patent 8,482,403.