Fun Hat

Allison Z. Durkan
Hannah A. Hebert
Elizabeth R. Kenny

EC450 Final Project
Spring 2016

Goals

The objective of this collaborative enterprise was to create a project integrated with an MSP430 in order to demonstrate the skills and methods gained throughout the semester. The proposal by the "Fun Hat" group was to construct a system involving a string of LEDs and a speaker that are controlled by the MSP430. The MSP430 would also be connected to a Bluetooth peripheral, allowing it to communicate with other integrated Bluetooth system. Another goal was to build an Android application that sent information to the Bluetooth peripheral in order to change the state of the system.

Design

The first part of the design involved an Android application that connects to the Bluetooth module. That meant the app needed to utilize the Android Java libraries that define how Android devices interact via Bluetooth. Since the app was used to transmit information to the MSP430, not to receive, the application was only responsible for connecting to the Bluetooth peripheral and writing data bytes to it. These written bytes are what decided the state of the MSP430, which controlled the behavior of the LEDs and the speaker.

In order to construct the fully functional Fun Hat, the chip had to come off the MSP430 board. It was put onto a breadboard and all the connections, to the Bluetooth peripheral, the LEDs, and the speaker, were connected to the buses on the breadboard that corresponded to the correct pins from the chip. The LEDs were pushed through the interweaved straw on the hat, which helped stabilize and keep them in place. The speaker was pushed through the lining and stabilized with tape. The Bluetooth peripheral was placed inside the hat lining as well, but on the side so the user would be able to see, via the flashing light on the module, whether it was connected.

Description of Implementation

For the Bluetooth connection from the Android application, the MainActivity.java file on load of the app created an interface that had a list of the devices that had been paired with the phone the app is running on. The user can then select which of the paired devices the phone should communicate with by pressing the list item. On press, a thread connection between the phone and the selected Bluetooth device is established. If this fails, the app updates the text status on the interface to tell the user that it was unsuccessful and the exception that was thrown upon failure. If it succeeds in establishing the connection thread, a toast pop up with the description of the device it is connected to and the buttons to select a level of obnoxious appear. The text status is updated to a success message and the name of the device. Then, when a user

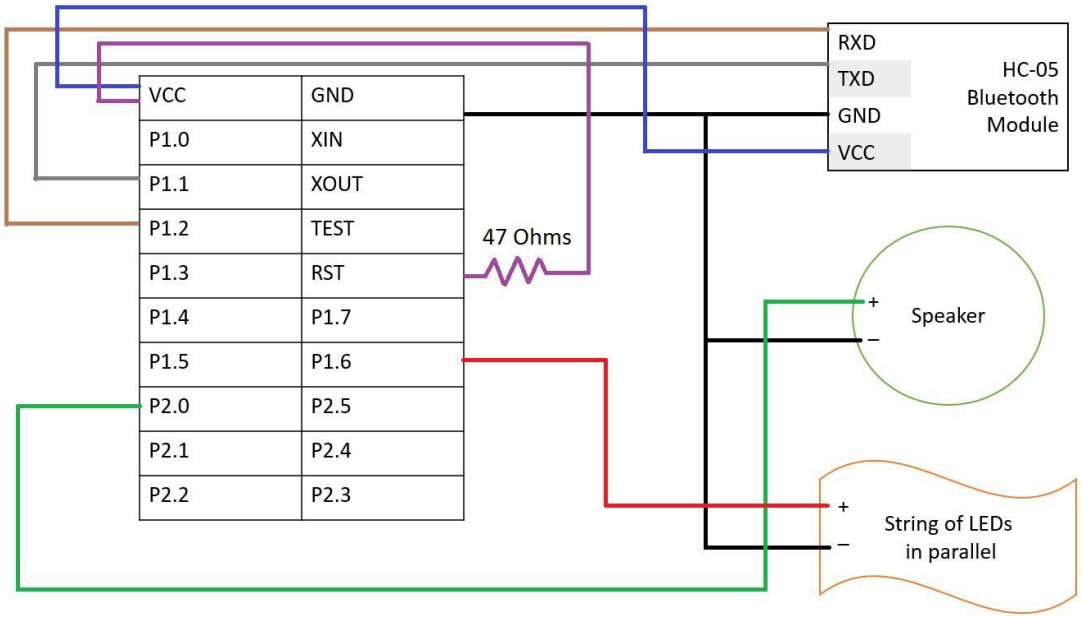
selects an obnoxious level, it transmits a character. The character to transmit is converted to bytes and sent through the connected thread to the receiving device.

To implement the receiving of characters, a global variable called Rx_data that is updated each time the UART receive interrupt is called. This variable is then used as a conditional in a switch statement that is continuously checked in main. The variable stores one character byte that is sent through the android application and received by the bluetooth module. The possible character bytes include 'O' for off, 'L' for "low key", 'M' for "medium key", and 'H' for "high key". To implement the LED blinking patterns, the Timer A0 channel was used on pin P1.6. The blinking pattern was determined by a global flag variable called fade_flag.When the timer A0 interrupt is enabled, it checks the fade_flag condition and either gradually varies the half period and updates the TA0CCR0 value, gradually fading the LED using the concepts of pulse width modulation, or, toggles P1.6 based on a counter variable, blinking the LED. In case 'O', all interrupts are turned off as well as P1.6 in order to halt both the music and the lights. In case 'L', the fade_flag was set to 0, the state was set to 1, and the Timer A0 and Timer A1 interrupts were both enabled, allowing the LEDs to toggle on and off while "Amazing Grace" is played through the speaker. In case 'M', the fade_flag was set to 1, the state was set to 2 and the Timer A0 and Timer A1 interrupts were both enabled. This enabled the hat to fade the lights in and out while "Mamma Mia" is played. In the final state "H", fade_flag was set to 1, TA0CCR0 was set to outmode 7, the state was set to 3, and both timer interrupts are enabled again. This caused the LEDs to blink quickly and sporadically while playing "Joy to the World".

The auditory aspects of this project were based on Professor Giles' use square wave tone outputs in OUTMOD_4 with the Timer A interrupt in the Tone 1 and Tone 4 examples. Each song was hard-coded using two arrays of integers: one array to indicate the frequency of the current note, and another of the same length to indicate the duration of that note. Only one octave of notes was necessary to hardcode the three songs that are outputted using pulse width modulation with the Timer A1 channel. Each frequency corresponds to a half-period value for the square wave output of the 1MHz clock used. When the Timer A1 interrupt is enabled, the interrupt uses global variables to count the notes in the constant integer arrays of frequencies and assign them to TA1CCR0 for the given period as defined by the corresponding element of that song's duration array. When a global int counter (period) exceeds the current note's duration, another global counter to track the current notes is incremented and the next frequency in the note array is added to the TA1CCR0 register. Using intuition and prior knowledge of music theory, the durations of each notes were chosen based on the desired maximum duration for the given state of the hat, and other durations were then selected with respect to that value based on the indications of the sheet music. Also in the interrupt, the size of the present state's note array is called and compared to the current value of the note count variable. If the note count index is

about to exceed the size of the song, the counter is reset to zero and the song is restarted. The current song will continue to play on repeat until the state is changed.

Schematics



Assessment of Success

Overall, the Fun Hat was an overwhelming success. After the proposal submission and the meeting with the TAs to discuss it, the group was definitely nervous about meeting the goals outlined by the updated proposal. However, after hard work and a lot of bumps in the road, the group was able to reach and complete all tasks. The LEDs and speaker are controlled by one chip, which has been taken off the MSP430 board. The chip is also connected to a Bluetooth module which connects to the Android application that was built, and this app is able to successfully select the state of the system. The Fun Hat works with a power source that can supply an ample amount of voltage to power the entire system, which is not limited to a computer or a power bank.

Steps to Improve Success of Design

Most of the design limitations occurred in the assembly of the circuitry within the fedora. After the initial choice for an LED string was incompatible with the MSP430 power supply, the choice was made to instead create a custom string of LEDs, soldered in parallel to one wire connected to VCC and one wire connected to ground. Due to our limited experience with soldering, some of the connection points were not ideal. There were several points in which the conductive portions of the wire were exposed, causing LED malfunction if the VCC and ground

wires crossed. In order to improve this design, we were able to effectively wrap these bare wire points in electrical tape, preventing any shorts in the circuit.

Once the logic and circuitry was working properly, the next step in improving the design was to allow the circuit to function independently of the microcontroller. In order to do this all pins were connected as they were on the MSP430 but with the addition of a battery pack of two C-batteries connected to VCC and a 1.1 kΩ pull-up resister added between the reset pin and VCC. While this allowed the project to work independently of the MSP430, powering the device without the controlled source of the launchpad proved to be difficult.

Limitations

Although the Fun Hat device successfully communicates with the Android app via Bluetooth and changes states as necessary, it seems that the circuit consumes power at rates greater than anticipated. Two new C-batteries in series provide the 3.2V necessary to power the HC-05 module, the circuits, and lights while successfully changing state with user input; however, after relatively short periods of use the quality of the battery pack seems to diminish. After probing power source provided by the two C-batteries, it was seen that a current of about 4A was provided to the circuit. This is about 20 times more current than would be provided through a power bank connected with the MSP430 which would normally provide the circuit with 200mA and 3.3V while operating without issue. The HC-05 accepts voltages between 3.15-6V, so the excess current from the batteries was seems to trigger an automatic shut-off in the module to prevent damage.

Code and Listing

Please see the files submitted in the zip folder.

Summary of Team Contributions

The team, as it was composed of three members, split up the workload three ways according to the three main components of the system. Allison was in charge of designing and programming the Android application and making sure it could send the correct information to the hat. Hannah worked on implementing the LED functionality, utilizing the Timer A0 interrupt to vary brightness as well as the UART character reception logic. Elizabeth made sure that all sound functionality through use of Timer A1 to modulate the frequencies of the square waves used to produce tones was effective and efficient. Elizabeth and Hannah also worked to build the LED string and implement the hardware. The whole team helped to put all the system components together and do the final assembly of the Fun Hat.