EEL 4924 Electrical Engineering Design 2
(Senior Design)

Final Report
19 April 2011

Project Name:
Twisted Text

Team Members:
Kyle Lewis <ktlewis02@gmail.com>
Mike Ferlazzo <mferlazzo@gmail.com>

Project Abstract:
Our project for consists of building a small desktop game based on the online flash game “Text Twist.” There are five individual units each with a microprocessor for control, a display, a button, a piezo speaker, and an IR communication system. The game starts when each unit is lined up and woken up with a button press. One letter is displayed on each unit and the user has to line up the blocks in different combinations to spell 3 to 5 letter words. After some time limit the players score is displayed out of a maximum and the player can start a new round.

Table of Contents

List of Tables

List of Figures

Introduction

The application domain for this product is the consumer entertainment and potentially student education.
This project is based off of a product currently on the market called “Scrabble Flash.” I was actually amazed at how it worked and thought it would be fun to build.

Product Features

Since this is a consumer electronics toy we want Twisted Text to be cheap and user friendly. The user should never feel frustrated when playing because his word does not register. Twisted Text will inform the player when he has spelled a correct word or if he is trying to spell something that has already registered.

Portability and battery life are pivotal in any consumer electronics game. The final block prototype would ideally be small enough so that five of them can be carried or stored easily in a purse or glove box. The battery life should be long enough for something like a family vacation and ideally last for months if not years of casual use.

Technical Objectives

Battery Life:
Each block needs a speaker and a display. Displays can potentially draw a lot of power so we selected a low power cell phone screen and tiny piezo speaker.
The MSP430 line of microcontrollers is marketed as being ultra low power. The software always reverts to a “Low Power Mode” or LPM while idling. There is also a mechanical switch to ensure the block is off.

Wireless Communication:
The main challenge is in the communication between each unit to determine which word is being spelled. We chose infrared (IR) technology because of the large amount of resources on the subject in addition to its low cost.

Competitive Products

"Text Twist" is an online Java based game available at Yahoo.com and the inspiration of Scrabble Flash. At the start of the game you are given a set of six random letters. The point of the game is to spell as many real words with your random letters as possible within a short time limit.

Our project puts Text Twist into a table top format. Each letter is displayed on a "block" and the user physically manipulates by lining them up to form words.

The difficulty and dictionary increase greatly with each added letter so we will be using only five letters and two letter words are ignored.

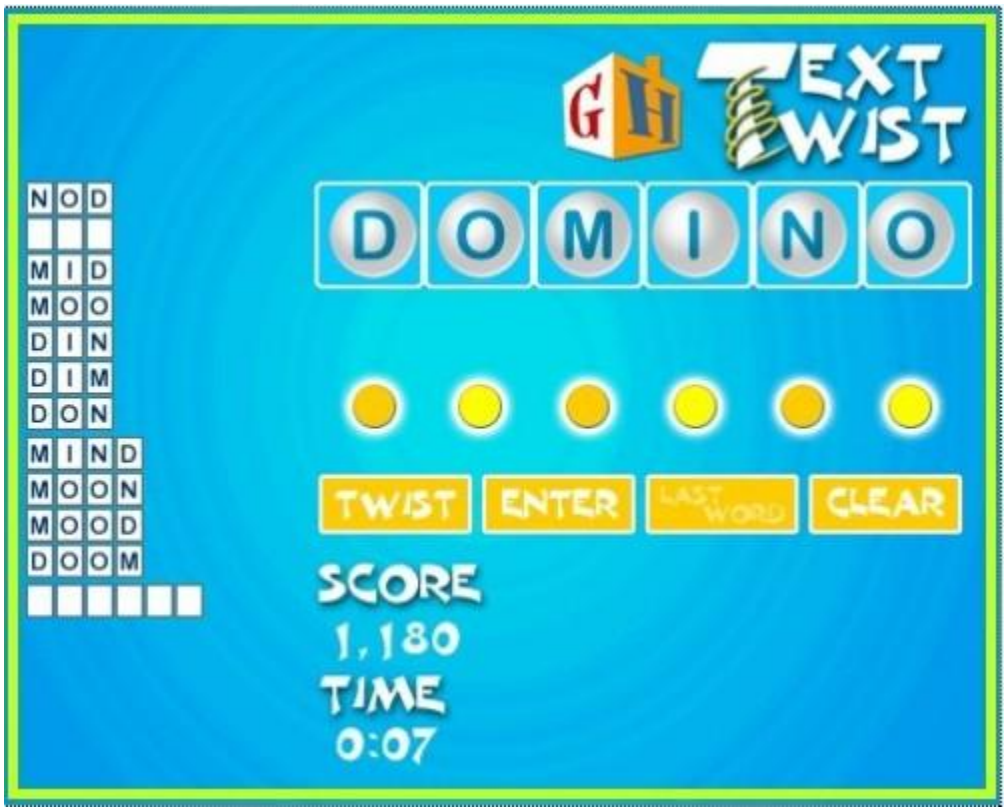


Figure 1: Text Twist in action. This user found the longest word!

Scrabble Flash is a table top version of Text Twist distributed by Hasbro. The game itself works great and there are very few causes for user frustration. Ideally Twisted Text will behave exactly as Scrabble Flash but realistically our project is just a prototype and we have very little experience with C coding and determining block order is complicated.

Figure 2: Scrabble Flash game.

Sifteo is another product dubbing itself 'The future of play.' But there is no actual product available at this time; there are only Youtube videos of prototypes. But what Sifteo appears to be is what Twisted Text could be in the future. A touch screen, color display, rechargeable batteries, USB and Bluetooth for PC connectivity, a faster processor and more memory, better sound, a developer API and 'App Store.' The possibilities are endless.

Figure 3: Digital blocks by Sifteo. This product has lots of potential.

Concepts and Theory

The main technical concept behind this project is infrared, or IR, communication. Luckily there is no shortage of information available on the theory behind the technology and its various protocols. The cheapest way to remotely control a device within a visible range is with IR light. Since we wanted a cheap wireless solution that relied on line of sight, IR was the obvious choice.

To ensure error free transmission, the signal is modulated so it stands out above the noise. With modulation we make the IR light source blink in a particular frequency. Common frequencies range from 30 kHz to 60 kHz in consumer electronics. The IR receiver will be tuned to that frequency, so it can ignore everything else.

IR Transmission:

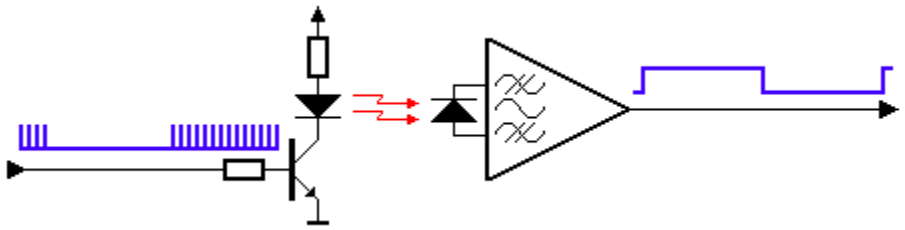


Figure 4: Presence of a modulated IR signal makes the voltage output of the receiver go LOW.

In serial communication we usually speak of 'marks' and 'spaces'. The 'space' is the default signal, which is the off state in the transmitter case. No IR light is emitted during the 'space' state. During the 'mark' state of the signal the IR light is pulsed on and off at a particular frequency.
At the receiver side a 'space' is represented by a high level of the receiver's output. A 'mark' (presence of modulated IR) is then automatically represented by a low level.
Note that the 'marks' and 'spaces' are not the 1-s and 0-s we want to transmit. The real relationship between the 'marks' and 'spaces' and the 1-s and 0-s depends on the protocol that's being used.

IR Reception:

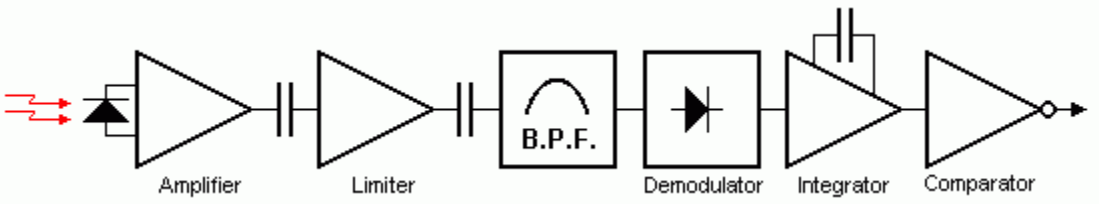


Figure 5: The block diagram of a typical TSOP IR receiver.

The received IR signal is picked up by the IR detection diode on the left side of the diagram. This signal is amplified and limited by the first 2 stages. The limiter acts as an AGC circuit to get a constant pulse

level, regardless of the distance to the handset.
As you can see only the AC signal is sent to the Band Pass Filter (DC blocking capacitor). The Band Pass Filter is tuned to the modulation frequency of the handset unit.
The next stages are a detector, integrator and comparator. The purpose of these three blocks is to detect the presence of the modulation frequency. If this modulation frequency is present the output of the comparator will be pulled low.

RC5 Protocol:

Phillips RC5 IR protocol is widely used and is considered easy to use among hobbyists. The protocol sends 14 bits total. Two start bits (S1, S0), one control bit (C), five address bits (A4 to A0), and a six bit command code (C5 to C0). The start bits are always '1' and the control bit toggles when a new command is sent. RC5 uses a carrier frequency of 36 kHz and has a constant bit time of 1.778ms (64 cycles of 36 kHz). The coding used is bi-phase or Manchester coding.
Manchester data is unique in that a data is signified by a transition in the middle of the bit. A one is transmitted as a space-to-mark transition and a zero as a mark-to-space transition. The entire 14-bit packet is received MSB first, starting with two start bits. The duration for the complete 14 bit packet is about 25 ms.

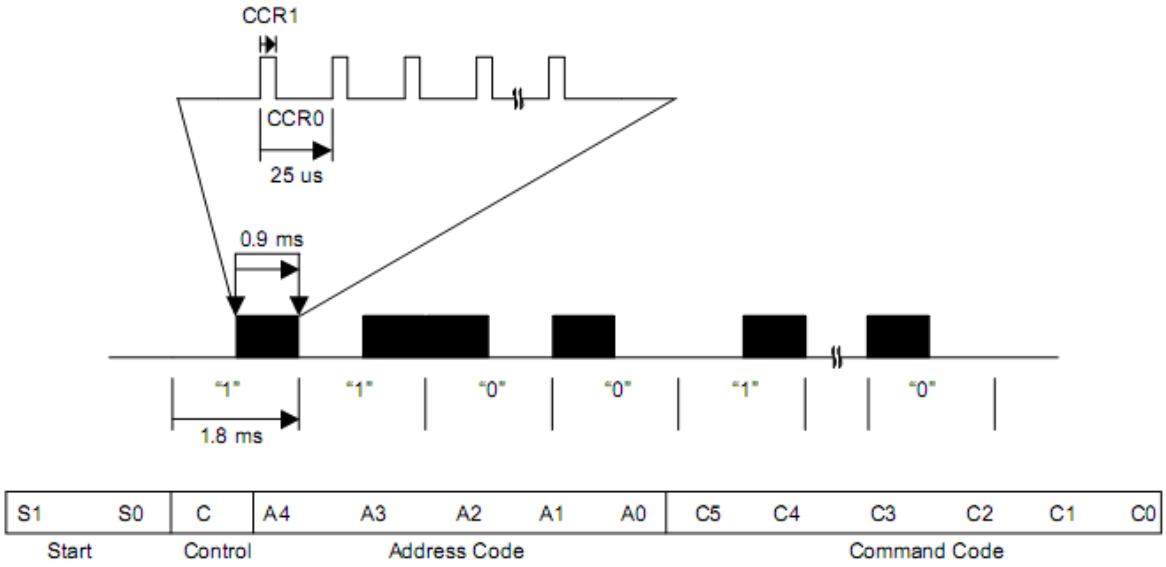


Figure 6: RC5 data packet example transmitted by the MSP430

Decoding IR Signals

- Method 1: Look at the signal level in time gaps of a bit's or half bit's duration. This gives a straight forward and simple implementation. However, the time intervals must be followed very strictly. Otherwise, synchronization will get lost and the receiver won't work. So one needs an exact timer. Moreover, not all RC5 transmitters follow exactly the RC5 signal standard and introduce tolerances in bit durations that make this approach obsolete.

- Method 2: Synchronize with the signal at each signal edge and evaluate the time that elapsed once the last edge. This approach is much more powerful: It does not need sophisticated timer capabilities, allows both integrity checks of the received data and allowing tolerances in the transmitted protocol.

Project Architecture

Microprocessor:

MSP430F2272 and 2274

- Low power and small footprint, tons of built in features
- Only difference between the two models is built in Operational Amplifiers in the 2274.
- Coded with C using Code Composer Studio v4 (gcc compiler)
- Stable oscillator without an external crystal

Figure 7: The MSP430F2272 IC

Display:

Nokia 5110 84*48 LCD

- basic graphic LCD screen with PCD8544 controller
- Serial interface and very low power consumption
- "The PCD8544 is a low power CMOS LCD controller/driver ... All necessary functions for the display are provided in a single chip, including on-chip generation of LCD supply and bias voltages, resulting in a minimum of external components and low power consumption. The PCD8544 interfaces to microcontrollers through a serial bus interface."

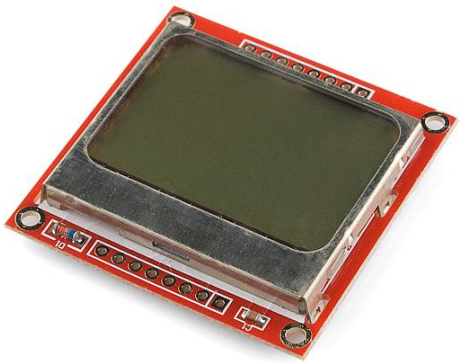


Figure 8: Nokia 5110 low power LCD

Wireless:

Transmission (TX) - IR Diode

- Circuit built as recommended by TI

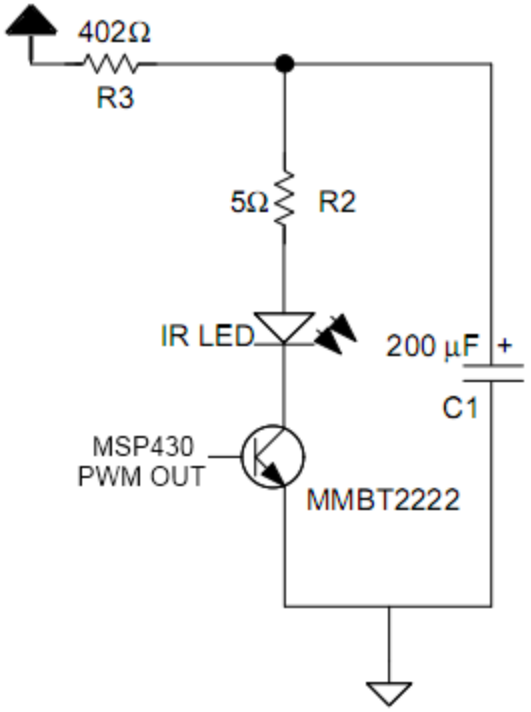


Figure 9: Recommend circuit for low power IR transmission.

Reception (RX) – Sharp D26x and Vishay TSOPxx36

- Work at 3V (most parts are 5V)
  - Sadly, Vishay TSOPxx38 do not work under 3.6V
- 3 pin package (GND, VCC, VOUT). Attach VOUT directly to a microcontroller input pin (with High to Low edge interrupt). Only a coupling capacitor is needed on the supply pins.
- amplify, filter, and demodulate the IR signal, providing a clean logic-level output with only the serial data present

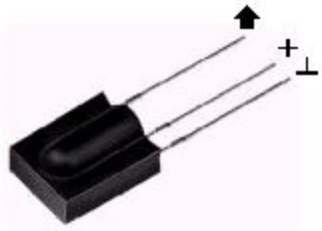


Figure 10: 3 pin TSOP package

Power:

3 volt CR2032 wrist-watch battery

- Perfect for MSP430 and recommended by TI for low power applications

Audio:

DAC and speaker plan was scrapped due to PCB size constraints and hassles with routing and vias. A lower power piezo speaker driven by PWM for beeps and blips is suitable for our prototype.

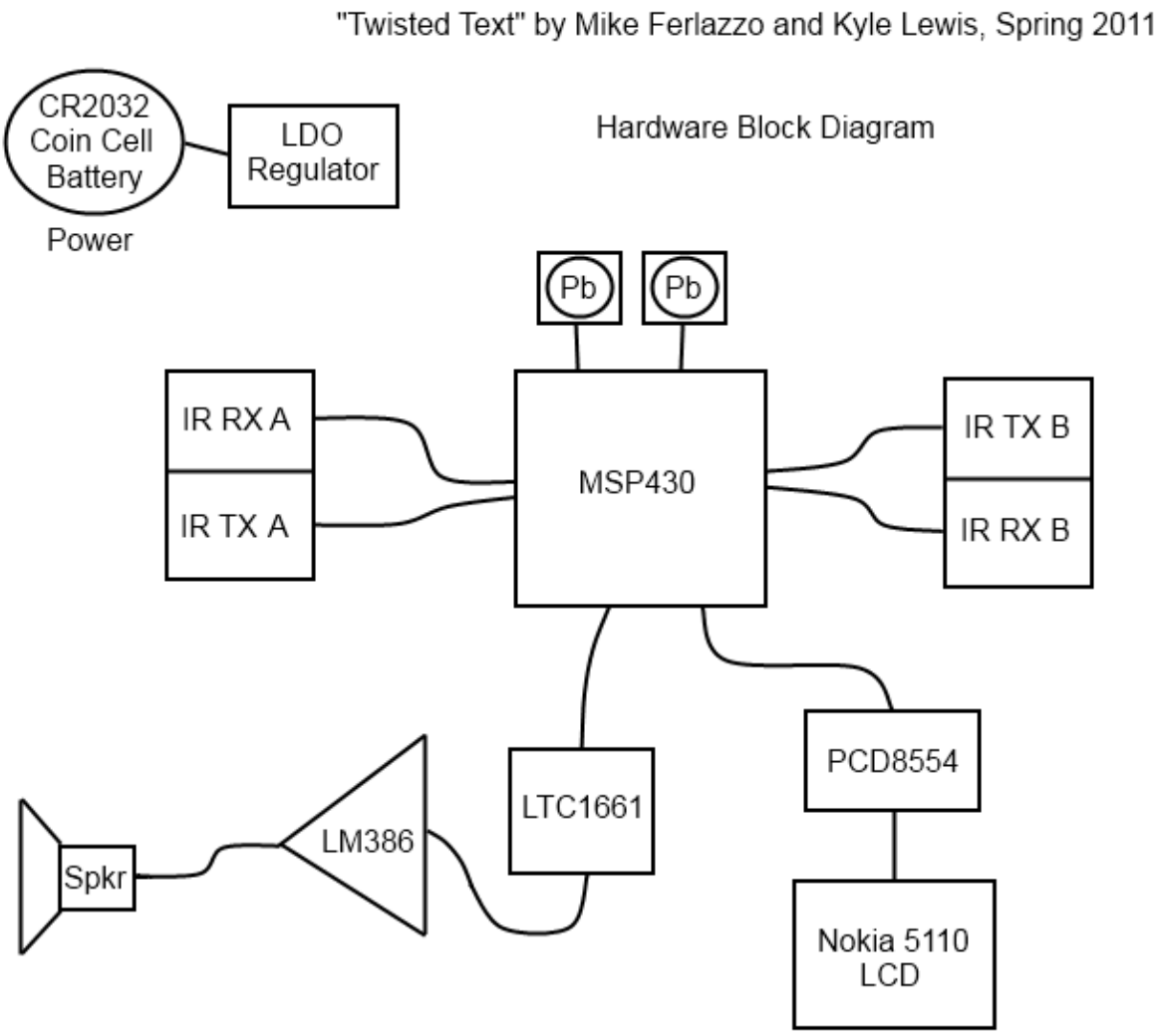
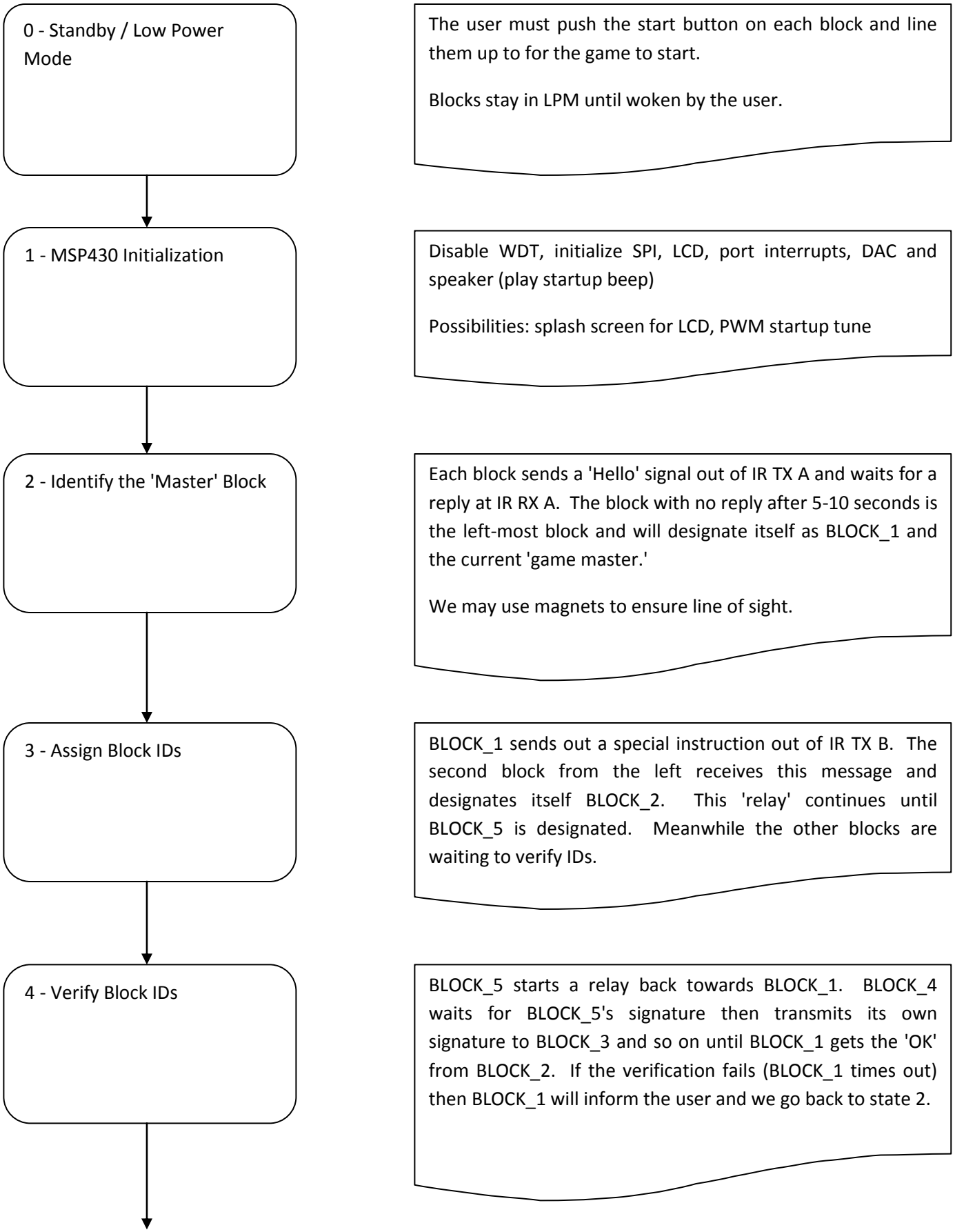
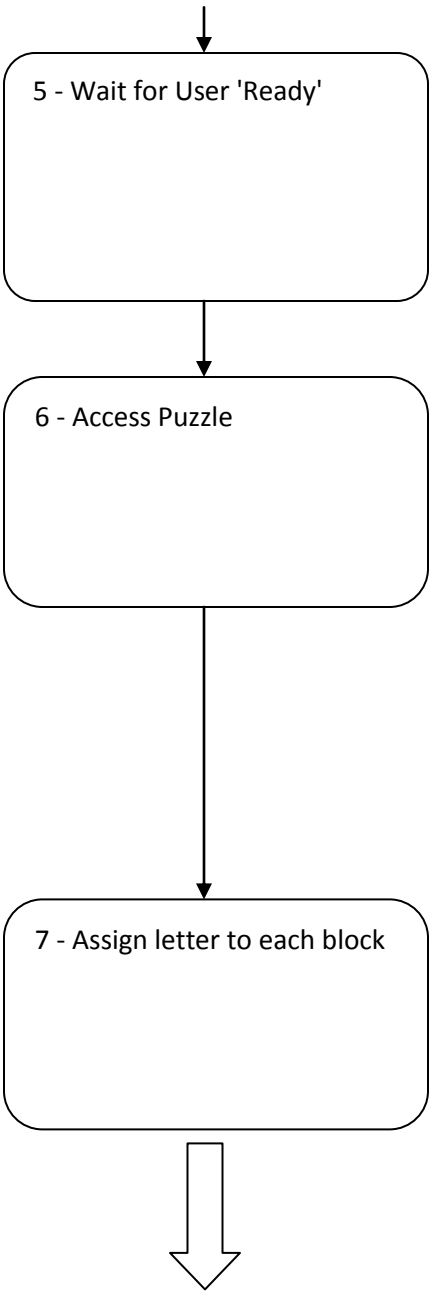


Figure 11: System Block Diagram

Software Flow Chart





LCDs display a 'ready!' screen and inform the user to push the button on BLOCK_1 to start the game.

Blocks will enter LPM after some length of inactivity (state 0).

BLOCK_1 fetches a puzzle from memory. Each puzzle will be stored in a 'dictionary.' The 'key' is the puzzle number (e.g. #000 - 999) and the first 5 items are the puzzle letters. The rest of the items are the solutions to the puzzle (3, 4 and 5 letter words).

Puzzles will ideally be chosen pseudo-randomly.

BLOCK_1 relays the puzzle # to each block then each block accesses the puzzle and assigns itself the item (letter) matching its block ID

The block displays the letter.

Game Logic:

Keeping up with block order seems like the biggest challenge. Keeping score is the next big issue. Because the blocks are always communicating with each other while falling in and out of an idle state and BLOCK_1 is not always in the puzzle solutions, each block will have to keep a list of correct answers. These lists must be consolidated at the end so BLOCK_1 (or the game master) can display the correct score at the end of the round. Words are not to be counted twice.

Determining the letter order:

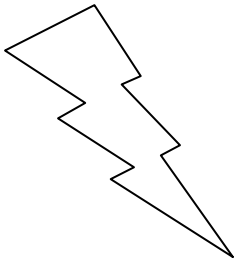
The idea here is similar to assigning block IDs. Blocks with no communication enter an idle state where they wait for an IR interrupt. The left most block is established then the right most block is established. The left most block designates itself the 'game master' for the moment. When the order of block IDs is established, the game master cross checks the users word with the items in the dictionary. If there is a match then the user receives audio and visual feedback. The solution is added to a solutions list which will be consolidated later.

Time's up!

When the game is over (90 seconds), the user is instructed to line up the blocks again. Then order is established and the list of solutions in each block are relayed to BLOCK_1 (or the left most block/current game master) and consolidated.

The user's score is displayed as correct answers out of total possible answers (which is the total number of items in the puzzle dictionary minus 5).

Solutions lists are cleared and the user is prompted to start a new game (back to state 6).



Note:

Any time a block is in an 'idle' state for longer than a minute or so it will revert back to low power mode (state 0).

Software Description

We first need a database in memory with several puzzles and their complete solutions. The main controller must keep track of the current puzzle being played and keep score all while managing the wireless communication.

The software operates like a state machine. The blocks are in sleep or wait mode until lined up and powered on by user via button (I/O interrupt). The block that is first woken up then wakes up the other blocks wirelessly via IR input interrupt. The blocks then wait again for the user to start the game. At the start of the game each block is assigned its 'random' letter to display. Then the blocks are always transmitting their personal block ID (0-4) and waiting for the ID of its neighbor(s).

A block with another block on one side is either a starting (left most) or ending (right most) block depending on which side the other block is on. Using the first (left most) block the letter order can be determined and we can check to see what the user is spelling.

If it is a correct word then the screens will flash and the speaker will chime and the users score increases by one (out of the maximum total solutions). If it is not a solution then there will be no user feedback to keep the prototype simpler.

The program must keep track of prior words spelled so as not to count them twice. If a word was spelled previously then we could flash the screen to remind the user but there will be no chime.

Bill of Materials

We did not expect our project to be expensive. The comparable commercial product has an MSRP of only $30 so although we are doing our own R&D we initially shot for under $300 (10x the product just to give us headroom). The final total came out to around $50 per block so we are slightly under our goal.

Cost Per Block:

MSP430F2272	Free
Nokia 5110 LCD	$10
IR LEDs	$1
IR receivers	$3
Block housing	$5
Piezo buzzer	$0.50
CR2032 battery + Housing	$4
Miscellaneous R, C, NPN	$1
Milling Cost + Shipping	$25
Total	$50

Table 1: Components and Pricing

Distribution of Labor

We wanted an even distribution of labor and in the end it seems to have worked out that way. Finishing on time would have been nice. Finishing earlier would have been nicer. But graduating is the most important thing to us.

Responsibility Table

	Kyle	Mike
MSP430 Programming	100	0
Parts Research, Ordering	0	100
IR Routines	75	25
LCD Routines	100	0
Speaker Routines	100	0
Game Logic	50	50
Housing Design, Construction	50	50
Altium Design	0	100
PCB Assembly	0	100
Final Assembly	50	50
Final Debugging	50	50
Presentation Board	50	50
Final Report	100	0

Table 2: Our responsibility table.

Timeline

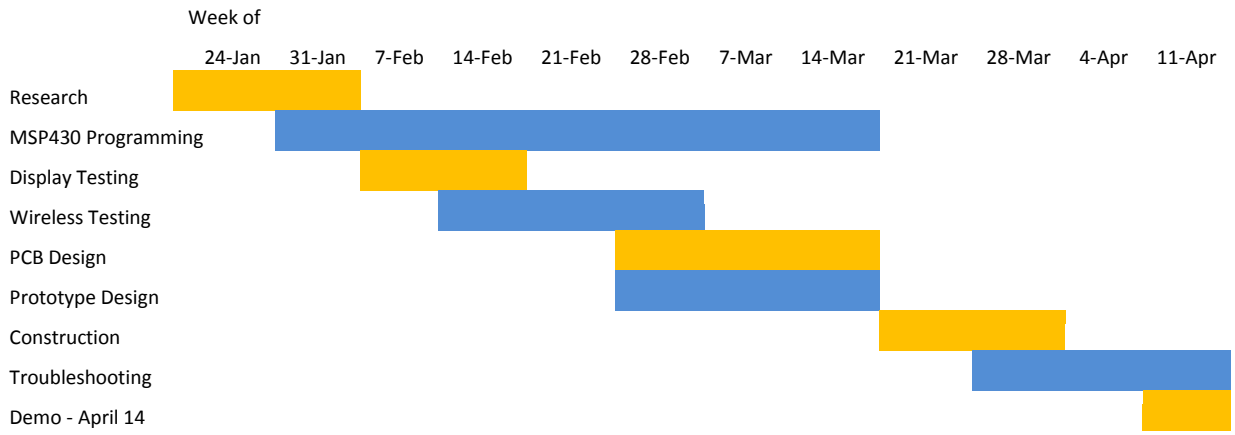


Figure 12: This Gantt chart shows how our project progressed (give or take 2 weeks; preferably give).

References

Miscellaneous:
EEL4924 Class Website <http://mil.ufl.edu/4924/>
Yahoo – Text Twist <http://games.yahoo.com/game/text-twist>
Sifteo Blocks – "The Future of Play" <https://www.sifteo.com/>

LCD:
Sparkfun – Nokia 5110 <http://www.sparkfun.com/products/10168>

CC Dharmani's Nokia 3110 C Routines
<http://www.avrfreaks.net/index.php?module=Freaks%20Files&func=viewFile&id=3446&showinfo=1>

Tinkerish – Modified 3110 Routines < http://tinkerish.com/blog/?p=50>

IR Theory:
SB Projects IR Knowledge Base <http://www.sbprojects.com/knowledge/ir/ir.htm>

RC5 Communication:
TI – MSP430 Application Notes:

- SLAA134 – Decode TV IR Remote Control Signals Using Timer_A3 <http://focus.ti.com.cn/cn/lit/an/slaa134/slaa134.pdf>
- SLLA175 – Low Power TV Remote Control Transmitter <http://www.ti.com/sc/docs/psheets/abstract/apps/slla175.htm>

Edaboard.com – RC5 Help Thread <http://www.edaboard.com/thread77201.html>

AVRFreaks – TWIRP
<http://www.avrfreaks.net/index.php?module=Freaks%20Academy&func=viewItem&item_type=project&item_id=616>

AVR – Atmel Application Notes

- AVR410: RC5 IR Remote Control Reciever - Atmel Corporation <www.atmel.com/atmel/acrobat/doc1473.pdf>
- AVR415: RC5 IR Remote Control Transmitter - Atmel Corporation <www.atmel.com/dyn/resources/prod_documents/doc2534.pdf >

PCB Schematic and Board Layout