ELEC 327: The Maze Runner - Electric Boogaloo
Written Report

Patrick Han
Chiraag Kaushik
Tommy Yuan

Overall Concept

The overall concept of this project was to create an interesting implementation of a classic toy where the object of the game is to navigate a small ball through a maze. We set out to implement the following features:

1. Motorize the tilt control such that manual control is not required
2. Control the motors using a wireless controller

Figure 1: The overall system

Components

We used the following components in our project:

Controller PCB:

- 1 RFM69HCW Transceiver Radio (433 MHz)
- 1 2-Axis Joystick
- 1 MSP430G2553
- 1 Battery Pack with 2 AA batteries

Board PCB:

- 1 RFM69HCW Transceiver Radio
- 2 MG90D High Torque Servo Motors
- 1 MSP430G2553
- 2 Battery Packs, each with 2 AA batteries

Mechanical Design:

- Plywood for the game board and frames
- Wooden dowels and string for rotation along each axis

System Architecture

Maze-controller Board:

The wireless transceiver is controlled using SPI. As a result, there were a few specific pins we need to connect to the MSP430. Firstly, the SCLK, MOSI and MISO lines were connected to the SPI pins corresponding to USCI module B0.

The other three pins CS (chip select), reset, and G0 are connected to generic GPIO pins on the MSP430. In particular, the G0 pin (GPIO 0) pin is used for interrupt request notifications from the radio. Thus it needs to be connected to an interrupt capable pin, which is the case for virtually all of the MSP430G2553's digital I/O.

Additionally, this device also drives two servo motors. These two motors (aside from VCC and ground) each require a "signal" pin that expects a PWM input. For this project we used timer modules TA0.1 and TA1.2 to generate the necessary PWM signals.

Finally, we have a switch and battery pack connected, along with all the other modules, to the 3.3 V VCC rail and ground.

A mapping of all the pins on the maze-controller MSP430:

Module Pin (Module)	MSP430G2553 Pin (Pin #)
RESET (TXRX)	P2.0 (8)
CS (TXRX)	P2.1 (9)
MOSI (TXRX)	UCB0SIMO (15)
MISO (TXRX)	UCB0SOMI (14)
G0 (TXRX)	P2.2 (10)
SCLK (TXRX)	UCB0CLK (7)
SERVO1 (Servo 1)	TA0.1 (19)
SERVO2 (Servo 2)	TA1.2 (12)

Controller:

The wireless transceiver configuration for the controller is the exact same as the maze-controller (the pin mappings are identical). Instead of servos, however, we have an analog joystick used for user input. The joystick is 2-axis and as a result requires 2 analog pins to input into. For our project we used analog inputs A0 and A3.

Pin mappings:

Module Pin (Module)	MSP430G2553 Pin (Pin #)
LR (Joystick)	A0 (2)
UD (Joystick)	A3 (5)

Code Architecture:

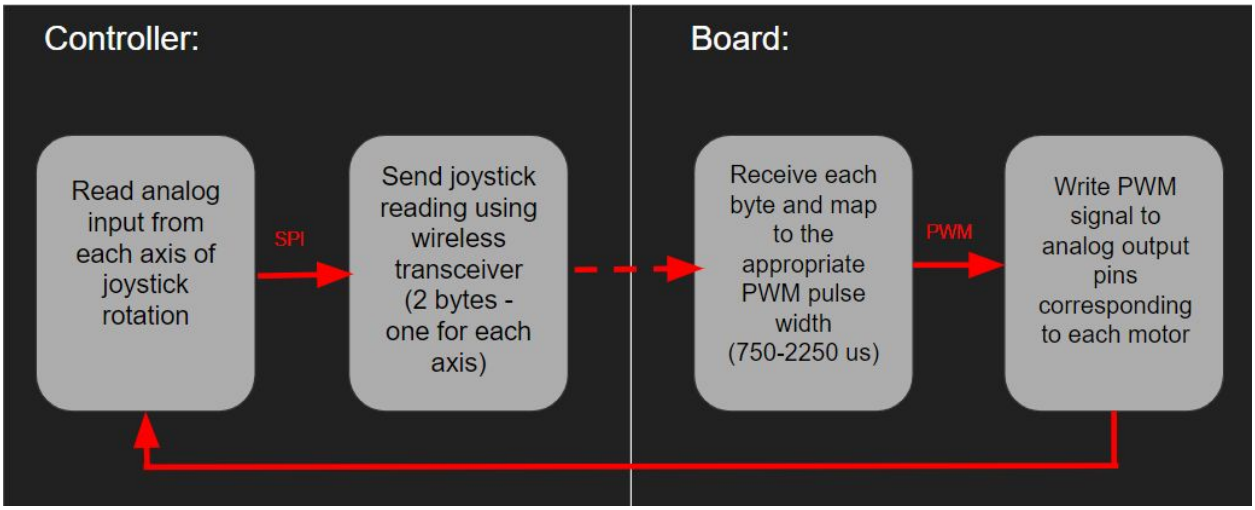

Figure 2: Code flow-chart

On the controller side, we first initialized the radio with an encryption key, the network number, and the address of the receiver radio. The radio is designed for use at 433 MHz, which is in the ISM (Industrial, Scientific, and Medical) frequency band, appropriate for remote control for our project. Then, we continuously read from pins A0 and A3 of the MSP, which are connecting the the L/R and U/D pins of the joystick, respectively. After obtaining two values, one for each of these axes, we use SPI to

communicate this data with the transceiver (using UCSI B0). For the transceiver, we use this library from LowPowerLab, which abstracts away the SPI communication and contains much of the functionality for wireless transmission. We had to modify the library to correspond with the chip select (CS) and interrupt (G0) pins used on our PCB. Once the transceiver receives the joystick readings, it maps each value to a single byte, for a total transmission size of 2 bytes. This transceiver is a packet radio which can send data in small packets of up to 61 bytes; however, we only required 2 bytes per packet for our project.

On the board side, after initializing the radio as above and initializing the servos to maintain a neutral (90 degree) angle, we wait for the interrupt which corresponds to a signal being received on the transceiver. Once the 2 bytes are received, each is mapped to a value from 750 to 2250, corresponding to the pulse width, in microseconds, of a PWM signal. These PWM signals are then sent to the appropriate servo (pins 12 or 19), causing the board to tilt as desired. For motor control, we used the Servo library included in the official release of Energia. This library uses Timer A0 to control the duty cycle of the PWM signal sent to each motor (via appropriate use of Timer A interrupts).

Mechanical Components:

The mechanical component of our project consists of a game board surrounded by two frames. The game board is 12 inches across, and each frame is roughly three quarters of an inch wider than the previous frame. Each frame around the game board allows rotation in one direction. Diagonal rotation is possible because the two frames operate independently and can be rotated simultaneously. We used the simulation capabilities of Solidworks to verify our design's dimensions and movement before we fabricated anything.

We made one prototype out of cardboard to verify the dimension tolerances and movement in real life. We then made the final version out of quarter-inch plywood. For both versions, we created Solidworks drawings of the required parts, saved them as Adobe Illustrator files, then cut the material using the Epilog laser cutter in the OEDK. The plywood version was additionally stained with mahogany coloring and coated in wood varnish to provide a nice, smooth aesthetic.

The servos control the rotation of the game board through a system of rods and strings. Each string is attached to opposite ends of the game board and wrapped around a rod. Rotating the rods increases the length of string on one side of the rod and decreases

the length of string on the other side. This creates an imbalance in the tension on either side of the rod, and the game board tilts in response. Multiple loops were made around each rod to prevent the strings from slipping during rotation. Each string is additionally tied to a rubber band to maintain tension and make the movements more responsive.

This relatively complex rod and string system was used instead of direct attachment to the frames because the servos do not have enough torque to turn the entire 12-inch frame. This system only requires the servos have enough torque to turn the rods, which have about an eighth inch radius and are mostly hollow.

Problems Faced

Of course, we faced a few problems while working on this project:

In our initial system setup, we only used a single battery to power both servo motors. After observing erratic system failure following a short period of normal operation, we decided to check the current draw on one of the lab Virtual Bench modules. Each motor when operating was drawing close to 0.25 A. Thinking this was a power issue, we installed a second battery pack specifically for driving one of the two motors. This fixed the problem and the system ran without hiccups afterwards.

We also faced some mechanical issues related to board rotation. Firstly, our servo motors were not able to achieve a full 180 degree swing. Secondly, the string and rubber band tension on the bottom of the board greatly affected the neutral starting or “zeroed” position of the board. If not properly adjusted, the board would be unable to rotate enough to move the ball in one of the directions. The fix to this problem was fine tuning the tension of the string on either side of the rod until the board was perfectly balanced, as all things should be.

Conclusion

We successfully achieved both goals stated at the beginning of this report. Not only are the features complete, but the controls are also very responsive. We hope others will take time to “run-the-maze” as well.