

EC450 Final Project — Team 8

William Chen, Anton Paquin

{chenwill, paquin}@bu.edu

1 Goal and Design

Our original project for this course involved emulating a GameCube controller for the purpose of performing tool-assisted superplay (TAS) routines on a GameCube or Wii game (for example, Super Smash Brothers Melee or Super Mario Sunshine). Unfortunately, we suffered a softbrick in one of the two MSP432 micro-processor boards at our disposal when attempting to set the main clock to its maximum speed (48MHz). Out of concern for breaking both boards and having no project as a result, we decided to abandon the aforementioned project and decided to venture into building a user-controllable miniature robot arm.

This robot arm was designed using five servo motors and had four joints; two drove a "shoulder" at the base, while three individual motors drove an elbow, wrist, and hand/claw. The MSP432 controlled actuation of these parts, and received commands from an internet-connected Raspberry Pi.

2 Implementation

From a system-level perspective, the three interacting hardware components involved the MSP432 micro-processor board, a robot arm built of five servo motors, and a Raspberry Pi 2. The MSP432 served as the interface that bridged user-defined inputs in software with the physical actuation of the robot arm. Hence, it connected to both the Raspberry Pi and the servo motors with the use of GPIO pins.

In terms of component-level peripherals used in the MSP432, we relied on the TimerA and the eUSCI SPI modules that drove the robot arm and communicated with the Raspberry Pi.

2.1 TimerA/Robot Arm

Five TimerA configurations were set up and linked to each GPIO output connected to the servos, and the modules were run in capture compare mode to toggle a high and low signal when the half duty-cycle period was reached, creating a PWM signal that drive the servos to a fixed position. The duty cycle period is calculated in the `servo_set()` function, which will set the appropriate compare value in the register (`halfPeriod-1`) when given an input from 0-180 (the range in degrees that the motors can move in).

3 Schematics

4 Outcome

The robot arm responded to inputs fairly accurately.

5 Improvements

While the servos were fairly accurate when calibrated, they were quite cheap and susceptible to heat problems and audible noise, potentially making them unsafe for long-term use. From the communications perspective, some control bytes and/or parity bits could be added to the protocol ensuring that the commands are always correct.

6 Code

All the code can be found on GitHub at

<https://github.com/antonpaquin/EC450-proj>

For the sake of convenience, the library routines and `main.c` are provided in the archive containing this report.

7 Human Resources

Anton was responsible for writing code involved with driving the individual servo motors as a library routine and debugged the finished product (calibration values).

Will was responsible for the communications interface between the MSP432 and the Raspberry Pi (SPI).