Homework 4:
Follow The Light Report

Submitted: November 27, 2017

Sabbir Ahmed

1 Description

This project utilized a mechanical servo mounted with a Light-Sensitive resistor to mechanically respond to a moving light.

The servo's movement range is about 180 degrees and is controlled with a PWM signal. The frequency should be around 125 Hz and the pulse width should correspond to approximately 6% to 30% duty cycle. The servo moves the arm in the range according to the width of the pulse.

The system is primary defined by two behaviors FULLSWEEP, LOCALSWEEP and two modes, FOLLOW THE LIGHT and AVOID THE LIGHT.

Modes

In one mode, Follow The Light, the objective is to find the angle of most illumination. The other mode, Avoid The Light, the objective is to find the angle of least illumination. The mode is set via a user menu using the onboard joystick and LCD. The mode should be displayed as `FTL` (follow the light) or `ATL` (avoid the light). Up and down button presses select the mode of operation. A right button press should initiate a full sweep.

Behaviors

In each mode, there are two behaviors: `FULLSWEEP` and `LOCALSWEEP`. The `FULLSWEEP` behavior is automatically followed by the `LOCALSWEEP` behavior. While in the `FULLSWEEP` behavior, the system should not respond to any button presses.

FULLSURVEY A full sweep of the range should be performed starting at 0° and sweep to 180° in about 20° increments. An illumination measurement should be taken at each step. The optimum angle (most light or least light depending on the mode) is decided and called the initial primary angle and is displayed on

the left 3 character positions on the LCD as a value in the range (0,180). The next behavior should then commence.

LOCALSWEEP Starting at the primary angle, the servo should continuously move in the cycle given below, finding the optimum of the three angles.

$$\textbf{primary angle} \rightarrow (\textbf{primary angle} - 10^\circ) \rightarrow (\textbf{primary angle} + 10^\circ)$$

After each cycle, a new primary angle should be decided and displayed. If the primary angle is at the limit of the servo range, the local search pattern will not involve one of the +10° or -10° measurements.

This document serves as the final report for the project. The report details the implementation of the project using C with `avr-gcc` and its development in a full Linux environment and the Atmel Studio development environment.

2 Implementation

The project was developed with a bottom-up design. Functions with more frequent usage and higher priority were developed first, and later pieced together to contribute to higher level functionalities. Because of reasons later detailed in the Troubleshooting section, the entire source code had to be contained in a single `main.c` script.

2.1 Light Sensor

The light sensor connected to PORTB utilized the ADC functionalities. The ADC value was used to determine the brightness of the light source, where more light corresponded to a smaller value whereas less light corresponded to a larger value. The value was used to control the direction of rotation of the servo mounted.

2.2 Memory Management

The implementation emphasized the limited memory on the chip. Usage of strings were minimized, and menu strings were stored as constant `PROGMEM` variables for reuse. The program memory was also utilized to communicate with the LCD with `lcd_puts_P` .

3 Usage

The project depends on user inputs from the Butterfly joystick. The program sits in a loop until the `RIGHT` joystick button is pressed. The user may press the `UP` or `DOWN` joystick buttons to toggle between the `FTL` and `ATL` modes. After a mode is selected, the servo begins sweeping with its corresponding configuration. The program terminates 10 seconds after a `LOCALSWEEP` followed by a `FULLSWEEP` are executed.

4 Testing and Troubleshooting

Debugging the functionalities of the program required extensive usage of hardware. This attribute resulted in numerous reprogramming of the AVR Dragon. Small changes, especially during experimentation of values of signals sent to external devices such as the servo, slowed down the development process.

The development process took place on a Linux machine since it was much more user-friendly and convenient that Atmel Studio. `avr- gcc` was used to compile the source code, and `avrdude` provided the AVR libraries. Once substantial progress was determined in the development, the source code would be pushed to the version control system to transfer to a ready-for-upload Windows machine to program the chip. Although this process proved to be a faster approach for development, it required all the changes to be contained on a

single file for frequent reprogramming.

Most of the issues arose from the external devices.

4.1 Light Sensor Issues

The light sensor was missing from the development kit, and therefore had to be replaced with another light sensor from a different vendor. The usage of the sensor still appeared identical, although there were frequent instances where it glitched.

4.2 JTAG Cable Issues

The JTAG cable from the kit does not work anymore. Temporary replacements were obtained from peers, which slowed down development further.

4.3 Servo Issues

The servo proved to be biggest obstacle during development. The kit included a DGservo 8g (blue) servo, which was later determined to be a continuous servo. Once a connection was established with the Dragon, the rotations seemed almost sporadic. Even with a timer integrated, the rotations did not appear to agree with the angles provided.

5 Code

The C scripts used for the implementation has been attached alongside the report.

5.1 main.c

The entire source code of the project.