Western Washington University
Electronics Engineering Technology

ETEC 471
Senior Project Description
Professor Morton

The Watchful Eye
Tyler Snow
12/10/03
INTRODUCTION

In today’s society, security is a big issue. Everywhere you look in public places these days are security cameras watching you from somewhere. The problem with the security cameras out there is that they are not very smart. The ones we use today are either controlled by a person in some security room, or they just pan back and forth and hope that they catch something on the screen. I propose to design a camera system that uses motion sensors to detect motion of a person, and aim the camera lens at that motion. This camera would provide a more reliable picture of any people moving in the room it is placed in. The camera would also have a manual override where a user in a security booth can manually control the camera if needed. I think this will be a fun project to work on and also is something that may be very useful. I will be designing the portion of the system which controls the camera. The camera itself will be a USB camera connected to a computer.

FUNCTIONAL DESCRIPTION

This camera system will incorporate four passive infrared (PIR) motion sensors to determine where motion is detected. These sensors will be checking the perimeter area. An explanation of how these sensors work is in figure 1. The sensors will be able to detect motion within at least 5 meters of the camera, so the camera would be able to handle a fairly large size room. The four sensors that I expect to use to search the perimeter are 10 meter range infrared motion detectors with a digital 5 volt output.
The output of these sensors will be fed into the HC12 microcontroller (MCU), and the microcontroller will determine where the camera should be aimed. The camera will then swivel towards the sensor or sensors that are active. When more than two sensors are activated at one time, the camera will automatically swivel back and forth 360 degrees. This would tell the camera that there is too much motion in the room for the microcontroller to determine what to do. The user could also manually select a mode where the camera will swivel back and forth instead of using the motion sensors. This will be done from a standard PC with terminal emulator software. There will be three modes altogether: Auto Find, Manual, and Auto Swivel.

The motion of the camera will be handled by a 5 volt bipolar stepper motor. Stepper motors are able to be controlled by applying voltages in a certain series to excite magnets internally. The way the motor is connected is shown in figure 2. Each step of the motor is measured in the degrees that it revolves. The smaller the step angle, the more precise the motor can rotate. The motor I am using has a 7.5° step. This small stepping degree will ensure good views with the camera.
Signals from the four outputs on the MCU will control the stepper motor. These signals will need to be buffered with a stepper motor driver chip in order to have enough energy to control the motor correctly and to be sure not to damage the MCU.

The motion sensors will be mounted in an enclosure that holds the MCU and all the other circuitry. The motor will stick out of the bottom of this enclosure with the camera attached. Figure 3 gives a good idea of what this will look like. The dimensions of the MCU enclosure will be limited to a 12” diameter X 6” high as seen in figure 4.
Hardware Description

MCU

For this system I will be using the M68HC912B32 (HC12) microcontroller. The microcontroller will be used to interface all the parts together and control the motor to aim the camera. I decided to use this MCU because of its versatility, and the fact that I have quite a bit of experience with it. There are also many resources available for this MCU in the ETEC 340 lab. I could have used an 8-bit MCU like the HC08, but I decided to use the 16 bit HC12 just in case I wanted to add new features later on in the development process. The resources I will use on this chip are shown in figure 5.

The HC12 will provide plenty of memory space for this project. This MCU includes 1kbyte of RAM, which will be used to store any temporary information like position and activity. It also includes 768 bytes of byte-erasable EEPROM, which allows the system to be turned off without losing vital information. The user interface of this project will use this EEPROM to store user password information. This MCU also has 32 kbytes of flash EEPROM, which is where the main program code will be stored.
I will be using the SCI port on the MCU to communicate with a standard PC. The PC will use a terminal emulator to communicate with the board. This will allow the user to control the position of the camera with the computer and select different camera options.

**Figure 5: Block Diagram**

**Motion Sensors**

As mentioned earlier, I will be using passive infrared type (PIR) motion sensors for this design. The sensors I picked are MP Motion Sensors from Matsushita, nicknamed NaPiOn. These sensors are the smallest of their type which measure .75 inches long X .68 inches in diameter. They include built in amplifier circuitry and a built
in voltage comparator. This makes these sensors very easy to interface with the inputs of the MCU, because the fact that the output voltage will be a 5 volt digital CMOS compatible signal. I chose to use a 10 meter type sensor to be able to sense motion in a large size room. The circuitry is contained in a T05 metal package which provides great noise withstanding capability. I need these sensors to be able to withstand noise because they will be mounted fairly close to the motor. I will use twisted pair wiring to help shield noise from the output of the sensors to the MCU.

Stepper Motor

As mentioned earlier, the stepper motor I decided to use has a step angle of 7.5°. This is plenty accurate to have a good range of sight with the camera. The camera can see a whole room with a 45 degree step angle in most rooms. When the motion detectors are on and detecting the angle, I will use will be 45 degrees, or 6 motor steps. This will let the camera face directly between two active sensors. The motor will be able to be controlled in clockwise and counter clockwise rotation. In order to drive this motor I will need to use a stepper motor driver chip, which I chose the L293D. This chip includes built in current limiting diodes. This chip accepts a TTL signal input, so I will use buffers on the output of the MCU to ensure compatibility from CMOS to TTL. I will use the outputs from PortP of the MCU to control the motor.

Power Supply

To simplify circuitry, I made sure to find parts that all use 5 volts. This allows me to use a regulated 5 volt signal routed directly to each component. Since the maximum power requirements for the motor and motor driver alone equal up to about 10 watts, I found a good supply that puts out 2.5 amps. This will allow my system to dissipate 12.5
watts without overdriving the supply. The supply I found connects directly into the
120V-60Hz wall outlet, and has an internal regulation circuit. The power supply will be
mounted inside the MCU enclosure, so it could not be tampered with or unplugged.

**Position Detectors**

To feedback the position of the camera to the microcontroller I will use position
detector switches. I will use three of these switches which will relay the current position
back to the microcontroller when they are depressed. These will be depressed
automatically when the motor is in motion. This will make sure that the motor doesn’t
keep running in one direction, which would tangle the wiring of the camera.

**Camera**

The camera that will be used is a USB camera. USB cameras are fairly simple to
operate. It will be powered by the computer through the USB cable, so no other power
source will be needed. This camera will be connected directly to a computer to give a
display.

**Software Requirements**

The software code will be written mostly in the C language. I will learn how to write
code in C next quarter in ETEC 454. I am familiar with C++, which should make it an
easy transition. I may write some code in assembly depending on its complexity. This is
because sometimes it is easier to use assembly to get the code to work exactly how I
would like it to. Descriptions of the modules I will create are as follows:
### Module Description

**POSITION**  This module will initialize the motor to a certain position when first powered up. It will also make sure the motor doesn’t rotate in one direction too much by taking data from the position switches.

**SENSOR**  This module will check the state of the motion sensors, and relay them back to the microcontroller.

**KERNEL**  MicroC/OS will most likely be used to control the timing of system tasks.

**MOTOR**  This module will handle the control of the stepper motor. The speed and direction of the motor will be determined with this module.

**INPUT**  This module will handle the user input, display user interface, and handle unlock codes.

**TIMER**  This module will keep track of time for locking the user interface.

### User Interface

The user interface for this system will be fairly simple. As mention earlier, the camera will have three modes: Auto Find, Manual, and Auto Swivel. When initially plugged in, the camera system will start out in Auto Find mode. This way, if there is no computer attached, the camera will automatically do what it was designed to do.

If a computer is connected through the serial port a menu will be displayed under a terminal emulator. This will be initialized with the user first pressing the spacebar. When the user presses the spacebar, they must then enter a password to make any changes. The display will look similar to figure 6.
If the password they enter fails three times, the camera will be locked in Auto Find mode for 2 hours. Even if the camera is unplugged, the camera will stay locked once it is plugged back in. Examples are shown in figures 7 and 8.

When the user enters a valid password they will be given a menu with four choices. These choices are which one of the three modes they want to enter and an option to change the password. An example of what this menu will look like is in figure 9.
If the user selects Manual mode, a menu will be displayed on what to do. They will use the arrow keys on the keyboard to pan the camera back and forth. The camera will pan in 7.5° steps in this mode for great accuracy. The spacebar will again let the user change the mode. If the Auto Swivel mode is selected, the camera will automatically pan back and forth 360 degrees. The same password will be used to change the mode each time.

Whenever the user wants to change their password, they will first have to enter in the old password. The camera will lock again in Auto Find mode here if the user enters in the wrong password three times. If they enter the correct password, they will enter in their new password twice to verify it. The password will always be shown on the screen as asterisks and not the actual password they are typing in. An example of the password change screen is shown in figure 10.

**Figure 9: Menu Example**

**Figure 10: Password Change Screen**
A flow diagram of the user interface is shown in figure 11.

Figure 11: User Interface Flow Diagram

DEVELOPMENT PLAN

Schedule

Since the beginning of the fall quarter 2003, I have been figuring out the best way to implement the system, and have started ordering parts. I will do most of the work on the system in lab, and some at home. I will order the parts that I don’t have soon, and
also check out a MCU board to get started. There shouldn’t be any problems with getting parts on time, because I have many of the main ones needed. Most of the parts I don’t have can be replaced by a similar part with no problems. The following is a weekly schedule to show how I plan to complete the project by the spring quarter of 2004.

Winter Quarter 2004:

Week 1: Start testing motion sensors, and solder wiring. Research.

Week 2: Find useable circuitry enclosure. Attach and install motion sensors and motor.

Week 3: Order remaining parts. Complete enclosure.

Week 4: Test motor control chip with motor.

Week 5: Attach position switches to motor assembly.

Week 6: Design noise filters to combat motor noise.

Week 7: Start software design.

Week 8: Start MicroC/OS kernel.

Week 9: Start on Motor module.

Week 10: Complete Motor module.

Spring Quarter 2004:

Week 1: Complete MicroC/OS module.

Week 2: Complete Timer module.

Week 3: Complete user interface. Hardware review.

Week 4: Complete Sensor module. Complete hardware documents.

Week 5: Testing and debugging.

Week 6: Integrate all software into prototype.
Week 7: Testing and debugging.

Week 8: Prototype testing.

Week 9: Code Review.

Week 10: Dead Week.

If I follow this weekly schedule I should have no problem completing the project on time. I need to make sure that if I get caught up on one part for a certain amount of time, that I make up for the lost time the next week.

Development Hardware and Software

Development of the software will take place mainly in the ETEC 340 lab. The hardware I will need to develop this project include: a digital multimeter (DMM), a mixed signal oscilloscope, a digital oscilloscope, a soldering iron, and some other tools that I have at home to create the casing. The development software that I will use to help create the software are the CodeWright system and the Noral debugger. Debugging will be done thorough the BDM port on the MCU.

Demonstration

I will demonstrate my project at the end of the spring quarter in the ETEC 340 lab. At this time I figure that I will have all the circuitry enclosed in the MCU case. I will plan to temporarily mount the system on the ceiling in the lab, without damaging the building itself. The lab should be a perfect size room to demonstrate the range of the motion detectors.

I will demonstrate all three modes of the camera, and also change the password in front of the class. The demonstration will end when I select the wrong password three
times, and the camera enters its locked state. I will then disconnect the power and reconnect it to show that the camera will stay in this state.

For the display I plan to bring in my laptop PC which has the drivers installed for the camera. This way I will not need to install additional drivers into the computers in the lab. Since my laptop doesn’t have a serial port I will use one of the computers in the lab to demonstrate the user interface. Terminal emulators are already installed on these computers, so this will be an easy connection.

For demonstration purposes only, I will run an extension cord to the power supply from the ground. In a real installation the power supply would be connected to an outlet or wiring above the ceiling grid.
## PROJECT SPECIFICATIONS

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Power Dissipation</td>
<td>10.33 W</td>
</tr>
<tr>
<td>Motor Speed</td>
<td>360° in &lt; 2 seconds</td>
</tr>
<tr>
<td>Power Supply</td>
<td>120V @ 60 Hz in, 5V out, 2.5A max</td>
</tr>
<tr>
<td>Enclosure Size</td>
<td>6” X 12” diameter</td>
</tr>
<tr>
<td>Total size with motor &amp; Camera</td>
<td>14” X 12” diameter</td>
</tr>
<tr>
<td>Operating Temperature</td>
<td>0°C to 40°C</td>
</tr>
<tr>
<td>Step angle</td>
<td>7.5° &lt; 360° (From Center Position)</td>
</tr>
<tr>
<td>Max Circuit board size</td>
<td>11” X 11” X 4” high</td>
</tr>
<tr>
<td>Reaction time to motion sensors</td>
<td>100mS</td>
</tr>
<tr>
<td>Minimum Sensor Range</td>
<td>10m</td>
</tr>
<tr>
<td>Lock Out Time Accuracy</td>
<td>2 hours ±- 10 minutes</td>
</tr>
</tbody>
</table>

**Serial Interface Specs**

- Interface: RS232
- Data Rate: 9600 baud
- Data Bits: 8 bits
- Parity: 0 bits
- Protocol: Asynchronous Start-Stop
<table>
<thead>
<tr>
<th>Part</th>
<th>Model Number</th>
<th>Source</th>
<th>Quantity</th>
<th>Price (Each)</th>
<th>Power Dissipation</th>
<th>Lead Times</th>
<th>Operating Temp. (°C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NaPiOn Motion Sensors</td>
<td>AMN14111</td>
<td>Arrow Electronics</td>
<td>4</td>
<td>15.40</td>
<td>1.2mA</td>
<td>Have</td>
<td>-20 to 60</td>
</tr>
<tr>
<td>Microcontroller</td>
<td>68HC912B32</td>
<td>Digikey</td>
<td>1</td>
<td>14.33</td>
<td>45mA</td>
<td>Borrow</td>
<td>0 to 70</td>
</tr>
<tr>
<td>Stepper Motor</td>
<td>35L048B1B-N</td>
<td>Digikey</td>
<td>1</td>
<td>10.37</td>
<td>1000mA</td>
<td>1 week</td>
<td>-20 to 70</td>
</tr>
<tr>
<td>Motor Driver Chip</td>
<td>L293D</td>
<td>Digikey</td>
<td>1</td>
<td>1.45</td>
<td>1000mA</td>
<td>1 week</td>
<td>0 to 70</td>
</tr>
<tr>
<td>Detector Switch</td>
<td>ESE-105HS1</td>
<td>Digikey</td>
<td>3</td>
<td>.48</td>
<td></td>
<td>1 week</td>
<td>-20 to 70</td>
</tr>
<tr>
<td>Regulated 5V Supply</td>
<td>DPS050240-P5P</td>
<td>Digikey</td>
<td>1</td>
<td>14.03</td>
<td></td>
<td>1 week</td>
<td>0 to 40</td>
</tr>
<tr>
<td>USB Camera</td>
<td>AC9801</td>
<td>CVOutlet</td>
<td>1</td>
<td>12.99</td>
<td></td>
<td>Have</td>
<td>Unknown</td>
</tr>
<tr>
<td>Quad Buffer</td>
<td>CD74HCT126M</td>
<td>Digikey</td>
<td>1</td>
<td>.14</td>
<td>.2mA</td>
<td>1 week</td>
<td>-55 to 125</td>
</tr>
<tr>
<td>USB Cable</td>
<td>CD74HCT126M</td>
<td>Radio Shack</td>
<td>1-10 feet</td>
<td>9.99</td>
<td></td>
<td>Have</td>
<td>Unknown</td>
</tr>
<tr>
<td>Twisted Pair Wire</td>
<td>ISW-6P24GY</td>
<td>Radio Shack</td>
<td>3 feet</td>
<td>.35</td>
<td></td>
<td>2 weeks</td>
<td>Unknown</td>
</tr>
<tr>
<td>Serial Cable</td>
<td>DB25 m/m</td>
<td>ABL</td>
<td>1-15 feet</td>
<td>9.99</td>
<td></td>
<td>2 weeks</td>
<td>Unknown</td>
</tr>
<tr>
<td>Resistors</td>
<td>Various</td>
<td>Mouser</td>
<td>10</td>
<td>.15</td>
<td>20mA</td>
<td>1 week</td>
<td>-55 to 155</td>
</tr>
<tr>
<td>Capacitors</td>
<td>Various</td>
<td>Mouser</td>
<td>5</td>
<td>1.00</td>
<td></td>
<td>1 week</td>
<td>-55 to 105</td>
</tr>
<tr>
<td>BDM Connector</td>
<td>87988-4</td>
<td>Digikey</td>
<td>1</td>
<td>.99</td>
<td></td>
<td>Borrow</td>
<td>Unknown</td>
</tr>
<tr>
<td>Reset Circuit</td>
<td>MAX6314</td>
<td>Digikey</td>
<td>1</td>
<td>1.04</td>
<td>.012mA</td>
<td>2 weeks</td>
<td>0 to 70</td>
</tr>
<tr>
<td>Crystal Oscillator</td>
<td>CA-103 16.0000M-C</td>
<td>Digikey</td>
<td>1</td>
<td>.96</td>
<td>.04mA</td>
<td>1 week</td>
<td>-20 to 70</td>
</tr>
<tr>
<td>RS232 interface</td>
<td>MAX232</td>
<td>Digikey</td>
<td>1</td>
<td>4.79</td>
<td>3mA</td>
<td>1 week</td>
<td>0 to 70</td>
</tr>
<tr>
<td>DB-9 Serial Connector</td>
<td>571-7478712</td>
<td>Mouser</td>
<td>1</td>
<td>3.66</td>
<td></td>
<td>1 week</td>
<td>Unknown</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td></td>
<td></td>
<td></td>
<td>$149.27</td>
<td>2,070mA</td>
<td></td>
<td>0 to 40</td>
</tr>
</tbody>
</table>