The Perfect Pint
Design Report

Image Courtesy of http://www.sapporo-guinness.co.jp.

Team “Guinness”

EEL4914C Senior Design Students:
Michelle Berecz [bereczgator@gmail.com]
Joseph P. Wilson [wilsonj1@ufl.edu]

Friday, December 1st, 2006
# Table of Contents

ABSTRACT .................................................................................................................. 4  
INTRODUCTION ........................................................................................................... 5  
TECHNICAL OBJECTIVES .......................................................................................... 6  
    COMPONENTS ........................................................................................................ 6  
    SETUP AND INTERFACING ..................................................................................... 7  
    ADDITIONAL FEATURES ....................................................................................... 8  
TIMELINE / GANTT CHART ....................................................................................... 9  
TASK DESIGNATIONS ............................................................................................... 10  
HOUSE OF QUALITY ................................................................................................. 11  
MATERIALS & PRICE LIST ........................................................................................ 12  
APPENDIX A: PROBLEMS / SOLUTIONS / ISSUES / DESIGN CHANGES ...........13  
APPENDIX B: COMPUTER VISION ALGORITHM .................................................... 17  
APPENDIX C: PROTEL DESIGN SCHEMATICS AND FOOTPRINT ................. 20  
APPENDIX D: C CODE FOR OVERALL CONTROLLER SOFTWARE ................. 22
List of Figures and Tables

Figure 1: Kegmeter Setup ........................................................................................................... 5

Figure 2: The Perfect Pint Setup ............................................................................................... 6

Figure 3: Timeline / Gantt Chart .............................................................................................. 9

Table 1: Task Designations ....................................................................................................... 10

Table 2: Development Material List and Cost ......................................................................... 12

Table 3: Estimated Production Material List and Cost ............................................................ 12

Figure 4: Flow Chart for CV Algorithm .................................................................................... 17

Figure 5: Sample Image Sequence and Corresponding Difference Images ......................... 19

Figure 6: Protel Schematic ...................................................................................................... 20

Figure 7: Protel Footprint ......................................................................................................... 21

Copyright Guinness Corporation.
Abstract

Outside of Ireland, Guinness is treated like any other beer – pour and drink immediately. The proper way to serve a Guinness in the Irish manner involves a bit more time and effort! Incorporating an electronically controlled pump, microprocessor board, and CMU-cam to monitor the beer flow, The Perfect Pint aims to provide an automated system to serve Guinness in this age-old tradition. Computer vision algorithms, a microcontroller, and CMU-cam will monitor the Guinness level and send the appropriate control signals to the electronic pump. Ultimately, this system should pour “the perfect pint” even the Irish can enjoy. Slainte!
The Perfect Pint is more novelty than anything else. With this said, however, bars and pubs who cater to beer-connoisseurs may find this product a quick and easy way to pour an authentic pint of Guinness automatically. Electronic beer dispensers are not new to the market; the Kegmeter is a device that stores, chills, and serves beer from a keg. Using infrared sensors, cooling systems, and electronic pumps (among a plethora of other devices), the Kegmeter serves chilled beer one glass at a time. The Kegmeter dispenses the proper amount of beer into the glass based on volume. The main disadvantages of the Kegmeter are the gigantic physical size (as seen in Figure 1), expensive cost (nearly $5000), and lack of aesthetic appeal.

![Figure 1: Kegmeter setup](Image courtesy of http://www.kegmeter.com)

While the system prototype may not chill beer as the Kegmeter does, The Perfect Pint offers a much more compact and affordable beer delivery system. The Perfect Pint uses a visual method (as opposed to volumetric) to gauge the amount of Guinness contained within the glass. Our visual method allows for additional quality control features to be added. For instance, after the Guinness has been poured, the “head to body” ratio could be calculated; this ratio is a relative measure of the pour quality. The Perfect Pint could eventually be expanded to allow for various glass shapes, sizes, and beer types. At the moment, this system is limited to Guinness because of its distinctive separation of head and stout.
The proper method for pouring a Guinness is as follows:

1. Touch the tap to the side of the glass to reduce the amount of “head” in the beer.
2. Pour the Guinness to the Draught line on the official glass (approximately ¾ of a pint).
3. Wait 60 seconds for the Guinness to separate into a distinctive black stout and creamy head.
4. Top off the pint.

![Image of Guinness pour]

**Figure 2:** Rough sketch of The Perfect Pint system setup.

**Components**

- Standard Guinness glass & Guinness draught
- CMU-Cam
- “Roll ‘N Go” Electronic Pump
- Hardware components
  - PIC18F4320 microcontroller
  - External 8MHz Clock Oscillator (for more accurate timing)
  - 5V 2A Power supply
  - On/Off Switch
  - Debugging LEDs
  - Resistor pack (for LEDs)
  - Capacitors (to place in between power/ground)
  - IRF640 MOSFET
**Setup and Interfacing**

*Physical Configuration*

The standard Guinness glass will be placed on a flat surface beneath the tube connected to the electronic pump. A green-colored background will be placed behind the glass to provide contrasting color for image processing. This will allow the computer vision algorithm to detect the draught rising to the appropriate level. The camera will be positioned so that the majority of the glass is in the image; this camera position will be fixed. The power and Protel board will be integrated into the base of the system (in the “Roll ‘N Go” electronic pump base).

*System Process*

At a glance, the system will work as follows. A more detailed description is below.

1) Electronic signal starts pump and beer flows.
2) CMU-Cam takes a few pictures per second.
3) Computer vision algorithms analyze Guinness levels from CMU-Cam pictures.
4) At ¾ level, electronic signal closes valve.
5) Beer settles for 60 seconds.
6) Electronic signal opens valve until draught is topped off.
7) *Slainte!*

Upon pressing the start button, *The Perfect Pint* will start the pump, and let the beer flow into the Guinness glass. As the draught is pouring, the CMU-Cam will capture windowed images of the progress. These images will be processed by the PIC18F4320 microcontroller and analyzed using computer vision (CV) algorithms. More specifically, draught level will be found using a threshold - image subtraction method. Once the Guinness reaches the draught line of the glass as determined by the algorithms, the controller will send out a signal to halt the beer flow. An internal timer will count for 60 seconds, as the draught separates into stout and head. After this time, the controller will start to top off the Guinness glass. Using the same CV algorithms as before, the CMU-Cam images will be processed to determine the time at which the draught reaches the top of the glass; at this point, the controller will shut off the pump. The process is complete.
Additional Features for Later Versions of The Perfect Pint

- An LCD screen which displays phrases during the duration and completion of the pour.
- Using CV algorithms, the “head to body” ratio could be calculated after the pour is complete.
- A signature clover could be stamped/drawn on the head of the Guinness.
- Speaker with pre-recorded phrases throughout the progress of the pour.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact Bars re: funding.</td>
<td></td>
<td>24-30</td>
<td>1-8</td>
<td>9-15</td>
</tr>
<tr>
<td>Obtain &amp; Test PIC/MSP chip.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Run Camera.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Run Pump / Valve.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code basic Vision algorithms.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basic hardware/software interfacing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(without pump).</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test run algorithm.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interface &amp; test pump with system.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LCD integration.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fine tuning system.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Housing/aesthetics of system.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Key Dates**

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/15</td>
<td>Intermediate Status Design Report</td>
</tr>
<tr>
<td>12/1</td>
<td>Final Design Report</td>
</tr>
</tbody>
</table>
### Task Designations

<table>
<thead>
<tr>
<th>Task</th>
<th>Michelle Berecz</th>
<th>Joseph P. Wilson</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Hardware Interfacing</em></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><em>Microcontroller Interfacing</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Computer Vision Algorithm &amp; Software Development</em></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td><em>Pump Interface</em></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td><em>Project Funding Request</em></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><em>Housing / Aesthetics</em></td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

*Table 1: Task Designations for The Perfect Pint*
House of Quality

Tasks/Features ranked from most to least important:

1. Working Pump
   1. Integrated CMU-Cam
   1. Computer Vision Algorithm
   1. Interfaced microcontroller

2. Housing

3. LCD Screen

4. Head-to-Body Ratio Calculator

5. Progress Speaker

6. Clover Stamp
# Materials List

**Table 2: Total List and Cost for Development**

<table>
<thead>
<tr>
<th>Object</th>
<th>Date Purchased</th>
<th>Quantity</th>
<th>Price ($)</th>
<th>Total Cost ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMU-Cam 1</td>
<td>09.15.2006</td>
<td>1</td>
<td>$118.95</td>
<td>$118.95</td>
</tr>
<tr>
<td>¾” In-Line Sprinkler Valve <em>(Not Used)</em></td>
<td>09.18.2006</td>
<td>1</td>
<td>$17.43</td>
<td>$17.43</td>
</tr>
<tr>
<td>¾” pipe extension <em>(Not Used)</em></td>
<td>09.18.2006</td>
<td>1</td>
<td>$0.47</td>
<td>$0.47</td>
</tr>
<tr>
<td>Thread Sealing Tape</td>
<td>10.02.2006</td>
<td>1</td>
<td>$0.97</td>
<td>$0.97</td>
</tr>
<tr>
<td>¾” threaded adapter <em>(Not Used)</em></td>
<td>10.02.2006</td>
<td>1</td>
<td>$2.22</td>
<td>$2.22</td>
</tr>
<tr>
<td>½” CPVC Plug <em>(Not Used)</em></td>
<td>10.02.2006</td>
<td>1</td>
<td>$0.57</td>
<td>$0.57</td>
</tr>
<tr>
<td>5/8” Vinyl Tubing <em>(Not Used)</em></td>
<td>10.02.2006</td>
<td>2 feet</td>
<td>$0.36/ft</td>
<td>$0.72</td>
</tr>
<tr>
<td>PIC18F4320 Microcontroller</td>
<td>10.20.2006</td>
<td>2</td>
<td>$9.95</td>
<td>$19.90</td>
</tr>
<tr>
<td>8 MHz Oscillator</td>
<td>11.08.2006</td>
<td>2</td>
<td>$2.78</td>
<td>$5.56</td>
</tr>
<tr>
<td>IRF 640</td>
<td>11.15.2006</td>
<td>5</td>
<td>$1.69</td>
<td>$8.45</td>
</tr>
<tr>
<td>SPST Switch w/ green LED</td>
<td>11.15.2006</td>
<td>1</td>
<td>$3.21</td>
<td>$3.21</td>
</tr>
<tr>
<td>Wagner “Roll ‘N Go” Electronic Pump</td>
<td>11.15.2006</td>
<td>1</td>
<td>$40.00</td>
<td>$40.00</td>
</tr>
<tr>
<td>Mounting Brackets</td>
<td>11.25.2006</td>
<td>1</td>
<td>$2.78</td>
<td>$2.78</td>
</tr>
</tbody>
</table>

Total Cost of Development: **$221.23**

**Table 3: Estimated Production Price (per 1000 units)**

<table>
<thead>
<tr>
<th>Object</th>
<th>Quantity</th>
<th>Price ($)</th>
<th>Total Cost ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMU-Cam 1</td>
<td>1</td>
<td>$100.00</td>
<td>$100.00</td>
</tr>
<tr>
<td>Thread Sealing Tape</td>
<td>1</td>
<td>$0.97</td>
<td>$0.97</td>
</tr>
<tr>
<td>PIC18F4320 Microcontroller</td>
<td>1</td>
<td>$4.19</td>
<td>$4.19</td>
</tr>
<tr>
<td>8 MHz Oscillator</td>
<td>1</td>
<td>$2.78</td>
<td>$2.78</td>
</tr>
<tr>
<td>IRF 640</td>
<td>1</td>
<td>$1.69</td>
<td>$1.68</td>
</tr>
<tr>
<td>SPST Switch w/ green LED</td>
<td>1</td>
<td>$3.21</td>
<td>$3.21</td>
</tr>
<tr>
<td>Wagner “Roll ‘N Go” Electronic Pump</td>
<td>1</td>
<td>$30.00</td>
<td>$30.00</td>
</tr>
<tr>
<td>Mounting Brackets</td>
<td>1</td>
<td>$1.50</td>
<td>$1.50</td>
</tr>
</tbody>
</table>

Total Cost of Production: **$163.29**
Appendix A: Problems, Solutions, Issues and Design Changes

Microcontroller Change from TI MSP430 to PIC18F4320 with External 8MHz Oscillator

Because it is generally used for digital signal processing and image processing, the Texas Instruments’ MSP430 microcontroller seemed initially like a wise choice for a microcontroller. TI offered free software to program the MSP430 in C and even delivered free samples overnight. Unfortunately, even after milling a board for these surface-mount chips, the MSP430 was not the best choice. We had neglected to think about how to physically program this microcontroller; after searching for a programmer, we realized that we would have spent another hundred dollars. Because the computer vision algorithm used in The Perfect Pint was not computationally expensive, the PIC processor was an appropriate choice.

The PIC18F4320 offered numerous I/O ports, a built in RS232 port, SPI port, sizeable internal RAM and ROM, and C-programming functionality. Additionally, the Senior Design lab had PIC programmers throughout the entire lab, saving on the overall cost of The Perfect Pint. Because a stable clock speed was needed for baud rate generation (for communication with the CMU-Cam), we decided to use an external 8MHz oscillator. This oscillator allowed a stable, relatively fast baud rate of 38400 bits/second to be generated to communicate with the CMU-Cam. Additionally, the external oscillator created more accurate timing, especially when using the internal C command delay_ms(x), which creates a delay of x milliseconds. When using the internal oscillator, the delay_ms() function in C was not true to its timing; the external 8MHz oscillator solved this.

Grayscale Images Instead of Color Images

The CMU-Cam delivers pictures with three channels – red, green, and blue. Initialize, we were going to analyze each channel independently of each other. Nonetheless, the image-subtraction threshold algorithm we ultimately used relies solely on difference – regardless of what channel it is. To save on storage space and computation time, the three channels were combined into one by using a basic color to grayscale conversion (Grayscale = (Red + Green + Blue) / 3). As long as each object in the image (like the glass, background color, and beer) maintains distinct colors, the difference will be very apparent as the beer fills the glass. This grayscale image also allowed
us to use only on the internal RAM of the PIC18F4320 and not have to interface the external SPI memory.

*Image Subtraction Threshold Method rather than Sum of Square Differences CV Algorithm*

There exists a very robust computer vision object-tracking algorithm called Sum of Square Differences. From an initial point determined by the user (or computer), the algorithm can track the movement of an object by finding the smallest square difference between the current image and previous image using a small window area. This SSD algorithm searches throughout this entire window region to find the area most likely where the object one by minimizing the square difference. This SSD algorithm allows for object tracking even with a moving camera – a very robust algorithm indeed. Nonetheless, after initially implementing *The Perfect Pint* with the SSD algorithm, we realized that SSD is more computation than necessary for this project.

Because the CMU-Cam is stationary, the glass isn’t moving, and the only moving object is the draught level, a basic image-subtraction threshold method is much cheaper to implement. This algorithm (as described in multiple parts of this report) finds the magnitude of the difference between two consecutive windowed frames, determines if the pixel differences are significant using a predetermined threshold, and counts the number of significant difference pixels to determine if the appropriate level has been reached. This algorithm uses far less code, is faster, and works just as well for a stationary CMU-Cam.

*“Roll ‘N Go” Electronic Pump Instead of Electronically Actuated Sprinkler Valve*

Initially, we were using an electronically actuated sprinkler valve to act as the valve for the beer. As electrical engineers, we neglected the mechanics and physics of this situation; in order for the beer to actually pump through the system, there needs to be a pressure gradient. After trying numerous “home-made” fixes, such as trying to pressurize the beer before attaching it to the valve, we decided that the electronically actuated sprinkler valve was not a good option. We found an electronic pump used to pump paint from a can to a paint roller using tubing and a small piston. After making sure the “Roll ‘N Go” could be controlled by the PIC18F4320, we decided that this pump was the best option for *The Perfect Pint*. 
The preliminary design called for reading in and processing the CMU-Cam’s entire image (80x143 pixels x 3 color channels / image). We realized that the PIC18F4320 had only 512K of internal RAM, which was not enough to store three full-size images (the current image, the previous image, and the difference image). We ordered plenty of SPI memory, had the surface-mount SPI-memory milled onto a mini-board, and started trying to interface the memory. Nonetheless, the SPI memory interfacing was much more difficult than it seemed, even with extensive documentation. Because the CMU-Cam sends its pixel data over the serial port at 38400 baud, the speed of The Perfect Pint was severely limited by using the full image. As a space saver and time-reducer, a smaller window size for the images gathered from the CMU-Cam was chosen.

This smaller window size works in the overall scheme of the computer vision algorithm used. Because the ¾-level draught-line and the glass top are the only areas that The Perfect Pint cares about, a small window monitoring those areas would be ideal. This small window results in significantly less data sent over the RS232 communications port – the limiting factor in the speed of The Perfect Pint. Because fewer pixels are processed by the PIC18F4320, the response time is much better than with an entire image; with an entire image, the CMU-Cam would take nearly 3 seconds to transmit the image’s data. With the 50-pixel window area selected, the CMU-Cam takes fractions of a second to transmit all of the required data. Using the windowed area reduces the cost (by not using SPI memory), reduces the complexity (by not having to interface additional components), and speeds up the response time of The Perfect Pint. This was one of the best design changes we made.

CMU-Cam Communications Line Fried / Camera Processor Blown
The project was nearly complete, and we tried to hook up all the electronic components to a MOSFET switch. We grounded the TTL line of the camera to the source of the MOSFET, which either fried the communication lines on the camera or blew the camera’s internal microprocessor. The CMU-Cam wouldn’t even communicate with a computer through the serial port and the company-provided software. Regardless of what exactly happened, the CMU-Cam was unusable after this point.
With less than a week to complete the entire project, we had a momentary freak out session. Thankfully, a fellow student in Senior Design had a CMU-Cam2 from a previous project and was willing to let us borrow it for the rest of the project. We were going to purchase another CMU-Cam, but it would not have arrived in time for demonstration, so this was a saving grace. By switching to the CMU-Cam2, we had to change a small bit of the software interfacing of the camera with the PIC; however, by and large, the command set and interfacing was very similar to the CMU-Cam. In fact, the CMU-Cam2 provides higher image quality than the CMU-Cam, so this problem ended up being a blessing in disguise.

*Lighting Issues Affecting Product Performance*

Because the Guinness glass is clear, numerous reflections can be seen on the surface. This will continue to be a problem, because some of these reflections are in the location of the windowed images grabbed from the camera. As a result, the initial ¾-level draught line seemed much lower than it should have been; the reflection of the rising beer crossed over the windowed area, making the CV algorithm think it was the actual beer level. The best solution is to surround the area around the glass on all sides with a non-reflecting surface (like green construction paper); this should minimize the reflections on the glass. Additionally, we set higher difference thresholds to make it harder for a difference to be considered significant. These, in concert, should reduce problems due to reflections on the Guinness glass.

Additionally, the location of the external light affects the product performance by varying degrees. To minimize shadows and glare, the external light has been placed directly above the glass. Nonetheless, despite these modifications, the “top-off” level is slightly lower than it should be; however, this slight glitch makes sure the glass never over fills.
Appendix B: Computer Vision Algorithm

Figure 4: Flow Chart for Computer Vision Algorithm
Computer Vision Algorithm

The draught level will be determined using the computer vision algorithm outlined above. Because of space limitations, the entire frame will not be used; instead a smaller windowed image will be analyzed. This windowed image will be focused around the area of attention (such as the ¾-level draught line or the top of the Guinness glass).

The first windowed image (image_1) and second windowed image (image_2) will be read in from the CMU-Cam. The two images will be subtracted from each other, and the magnitude of their difference will be stored in a difference image array (diff_image). This difference array will be analyzed to determine whether the value is considered significant enough by comparing it to a threshold.

If the difference array value is greater than the threshold, the difference array value will be set to one; otherwise the difference array value will be set to zero. Because the camera is focused on a tiny area of interest, the number of difference pixels that are considered significant (diff_sum) is a relative measure of when the draught line has been reached.

If the number of significant difference pixels is greater than a defined threshold (determined by adjusting through initial experimentation), the draught line has reached the appropriate level, and the image analysis can stop. Otherwise, the algorithm will continue to analyze subsequent images until the number of significant different pixels has been reached.

This algorithm has worked on theoretical results (shown on the next page) and in experimentation. The theoretical results were used to initially test the validity of algorithm and used the entire image. The algorithm always assumes that the camera is stationary and the only thing changing is the beer level. On the following page, the difference frame shows the difference image between the current frame (on the same level) and the previous frame (the frame above). The difference frame essentially shows the location of the draft line, which confirms the validity of this algorithm.
<table>
<thead>
<tr>
<th>Image Sequence</th>
<th>Difference Image with Threshold Applied</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.jpg" alt="Image Sequence" /></td>
<td><img src="image2.jpg" alt="Difference Image" /></td>
</tr>
<tr>
<td><img src="image3.jpg" alt="Image Sequence" /></td>
<td><img src="image4.jpg" alt="Difference Image" /></td>
</tr>
<tr>
<td><img src="image5.jpg" alt="Image Sequence" /></td>
<td><img src="image6.jpg" alt="Difference Image" /></td>
</tr>
<tr>
<td><img src="image7.jpg" alt="Image Sequence" /></td>
<td><img src="image8.jpg" alt="Difference Image" /></td>
</tr>
<tr>
<td><img src="image9.jpg" alt="Image Sequence" /></td>
<td><img src="image10.jpg" alt="Difference Image" /></td>
</tr>
<tr>
<td><img src="image11.jpg" alt="Image Sequence" /></td>
<td><img src="image12.jpg" alt="Difference Image" /></td>
</tr>
<tr>
<td><img src="image13.jpg" alt="Image Sequence" /></td>
<td><img src="image14.jpg" alt="Difference Image" /></td>
</tr>
<tr>
<td><img src="image15.jpg" alt="Image Sequence" /></td>
<td><img src="image16.jpg" alt="Difference Image" /></td>
</tr>
</tbody>
</table>

No Difference Frame
Figure 5: Sequence of sample images with their corresponding difference images displaying the draught line level.

Appendix C: Protel Design Schematics

Figure 6: Protel Schematic for *The Perfect Pint*
Protel Footprint
Figure 7: Protel Footprint for The Perfect Pin
Appendix D: C Code for Overall Controller Software

/* Header file to control the Perfect Pint. */
/* perfect_pint.h */
/* Last Update: 11.28.2006 11:59PM */
/* Coder: Joseph P. Wilson */
/* Team Guinness */

#include <18F4320.h>
#device adc=8

/* Sets the various fuses for the PIC18F4320 */
#FUSES NOWDT    //No Watch Dog Timer
#FUSES WDT128   //Watch Dog Timer uses 1:128 Postscale
#FUSES EC_IO    //External clock
#FUSES NOFCMEN  //Fail-safe clock monitor disabled
#FUSES BROWNOUT //Reset when brownout detected
#FUSES BORV27   //Brownout reset at 2.7V
#FUSES NOPUT    //No Power Up Timer
#FUSES NOCPD    //No EE protection
#FUSES STVREN   //Stack full/underflow will cause reset
#FUSES NODEBUG  //No Debug mode for ICD
#FUSES NOLVP    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOWRT    //Program memory not write protected
#FUSES NOWRTD   //Data EEPROM not write protected
#FUSES IESO     //Internal External Switch Over mode enabled
#FUSES NOEBTR   //Memory not protected from table reads
#FUSES NOEBTRB  //Boot block not protected from table reads
#FUSES MCLR     //Master Clear pin enabled
#FUSES NOPROTECT //Code not protected from reading
#FUSES NOCPB    //No Boot Block code protection
#FUSES NOWRTB   //Boot block not write protected
#FUSES NOWRTC   //configuration not registers write protected
#FUSES NOPBADEN //PORTB pins are configured as digital I/O on
RESET

/* Sets up the 8MHz External Clock */
#use delay(clock=8000000)

/* Sets up the RS232 line for camera communications */
#use rs232(baud=38400,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8,TIMEOUT=1)

/* Defines window size of image taken by camera */
#define COLS 10
#define ROWS 5

/* Defines Threshold Values */
#define DIFF_THRESH 100 //Sets what difference is significant
#define DIFF_THRESH1 75 //Sets significant difference for top off
#define DIFF_SUM_THRESH 25 //Sets how many significant pixels needed
void main()
{
    char test[50];
    div_t idiv_r,idiv_g,idiv_b;
    int j=0, i=0, r, g, b;
    int stop=0;
    int im_x = 0, im_y = 0;
    int image_now[ROWS][COLS], image_prev[ROWS][COLS], image_diff[ROWS][COLS];
    int diff=0, pix_sum=0;
    /* *************************************************************/
    /* Variable Declarations */
    /* *************************************************************/
    /* Sets Various Internal Fuses on the PIC18F4320 */
    /* *************************************************************/
    setup_adc_ports(NO_ANALOGS|VSS_VDD);
    setup_adc(ADC_OFF|ADC_TAD_MUL_0);
    setup_psp(PSP_DISABLED);
    setup_spi(FALSE);
    setup_wdt(WDT_OFF);
    setup_timer_0(RTCC_INTERNAL);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DISABLED,0,1);
    setup_comparator(NC_NC_NC_NC);
    setup_vref(FALSE);
    setup_oscillator(False);
    /* *************************************************************/
    /* End Internal Fuse Setting */
    /* *************************************************************/
    /* *************************************************************/
    /* Sets whether Registers are Input or Output */
    /* *************************************************************/
    SET_TRIS_A(0b11111000); //sets Port A to all input except bit 2,1, and 0.
    SET_TRIS_B(0b00000000); //Sets Port B to all output
    SET_TRIS_C(0b01011111); //sets up Port C
    /* *************************************************************/
    /* End Register Setup */
    /* *************************************************************/
    /* Initialization phase to get the camera in sync */
    /* *************************************************************/
    delay_ms(200); // Gives an initial power up delay
for(j=0; j<5; j++)
{ printf("\r");

  // Gets ACK confirmation
  for(i=0; i<5; i++)
  { test[i] = getc(); }

  output_low(PIN_A0); // Debug LED Off
  output_b(0);
  delay_ms(500);

  // Checks that confirmation has been received
  for(i=0; i<5; i++)
  {
      output_b(0xAA);
  }

  output_high(PIN_A0); // Debug LED On
  delay_ms(400);
}
/* ************************* */
/* End Camera Sync Initialization Phase */
/* ************************* */
/* ************************* */
/* Image Analysis and Pump On */
/* Until 3/4 Draught Line */
/* ************************* */
/* ************************* */

// Sets the windowed image around the 3/4 level
printf("VW 29 71 %d %d", 28+COLS, 70+ROWS);
for(i=0; i<5; i++)
    test[i] = getc();

// Go on if window size set properly
    output_b(0xAF); // Output 0xAF to LED if sets window properly
    delay_ms(2000);

// Turns the electronic pump ON
output_high(PIN_A1);

// Send Frame #1
printf("SF\r");
for(i=0; i<5; i++)
    test[i] = getc();

// Grabs camera’s notification before pixel data
    output_b(0xFA); // Debug LED
    test[6] = getc(); // Gets new frame = 1;
    im_x = getc(); // Gets x size
    im_y = getc(); // Gets y size
// Grabs pixel data & converts to grayscale.
for(j=0;j<im_y;j++)
    for(i=0;i<im_x;i++)
        { idiv_r = div(getc(),3);
          idiv_g = div(getc(),3);
          idiv_b = div(getc(),3);
          r = idiv_r.quot;
          g = idiv_g.quot;
          b = idiv_b.quot;
          image_prev[j][i] = (r+g+b);
          output_b(image_prev[j][i]); // output image values to LED
        }
    test[9] = getc(); // gets new row
    test[10] = getc(); // gets end frame
    test[11] = getc(); // read in the colon at the end of DF

    /* // Debug LEDs to let know everything is OK after first frame gather
       output_b(0xAA);
       delay_ms(500);
       output_b(0x00);
       delay_ms(500);
       output_b(0xAA);
       delay_ms(500);
       output_b(0); */
    diff = 0;
    // Continually grab next frame and do analysis
    // until draught level is reached.
    while(!stop)
    {
        // Dump Next Frame
        printf("SF\r");
        for(i=0;i<4;i++)
            test[i] = getc();

        // Grabs camera's notification before pixel data
            { output_b(0xFA); // Debug LED
              test[6] = getc(); // gets new frame = 1;
              im_x = getc(); // gets x size
              im_y = getc(); // gets y size

              pix_sum = 0; // Used to determine if draught line reached

              // Grabs pixel data & converts to grayscale.
              for(j=0;j<im_y;j++)
                  { for(i=0;i<im_x;i++)
                      { idiv_r = div(getc(),3);
                        idiv_g = div(getc(),3);
                        idiv_b = div(getc(),3);
                        r = idiv_r.quot;
                        g = idiv_g.quot;
                        b = idiv_b.quot;
                        image_now[j][i] = (r+g+b);
                        output_b(image_now[j][i]); // output image values to LED
                      }
                  }
    }
```c
    test[7] = getc(); //gets new column
    test[8] = getc(); //gets end frame
    test[9] = getc(); //gets colon

    //Does Image Analysis
    for(j=0; j<ROWS; j++)
    {
        for(i=0; i<COLS; i++)
        {
            //Finds Magnitude of Pixel by Pixel Difference
            if(image_now[j][i]>image_prev[j][i])
                diff = image_now[j][i] - image_prev[j][i];
            else
                diff = image_prev[j][i] - image_now[j][i];

            //Sets Previous Image Pixel = Now Image Pixel
            image_prev[j][i] = image_now[j][i];

            //Determines if difference is significant
            //if significant, image_diff[j][i] = 1
            //else image_diff[j][i] = 0.
            if(diff>DIFF_THRESH)
                image_diff[j][i] = 1;
            else
                image_diff[j][i] = 0;

            //Keeps running total of pixels
            //that are considered significantly
            //different.
            pix_sum += image_diff[j][i];
        }
    }

    output_b(0x0F); //Debug LED

    //Checks if draught line is reached
    //by comparing the number of significant
    //difference pixels with a threshold
    //determined experimentally.
    //If the draught line is reached, then stop
    //taking more images and stop the pump.
    //Else continue with the frame grabbing
    //and image analysis.
    if(pix_sum>DIFF_SUM_THRESH)
    {
        output_b(0xFF);
        stop = 1;
    }
    else
    {
        output_b(0x01);
        stop = 0;
    }

    //Outputs the # of significant difference pixels
    output_b(pix_sum);
```

//Turns the electronic pump OFF
output_low(PIN_A1);

/∗ **************************** ∗/
/∗ Pump Off ∗/
/∗ 3/4 Draught Line Reached ∗/
/∗ **************************** ∗/

/∗ **************************** ∗/
/∗ Settle Mode ∗/
/∗ Wait 60 seconds ∗/
/∗ **************************** ∗/
for(i=0;i<60;i++)
{ output_b(0xAA); //Debug LED
delay_ms(500);
output_b(0x00); //Debug LED
delay_ms(500);
}
/∗ **************************** ∗/
/∗ End Settle Mode ∗/
/∗ **************************** ∗/

/∗ **************************** ∗/
/∗ Image Analysis and Pump On ∗/
/∗ Until Beer Topped Off ∗/
/∗ **************************** ∗/

//Sets the windowed image
//Image is higher up to check
//if top off point is reached.
printf("VW 39 1 %d %d \r",38+COLS,0+ROWS);
for(i=0;i<5;i++)
test[i] = getc();

//Go on if window size set properly
{ output_b(0xAF); //output 0xAF to LED if sets window properly
delay_ms(2000);
//Turns the electronic pump ON
output_high(PIN_A1);

//Dump First Frame
printf("SF\r");
for(i=0;i<4;i++)
test[i] = getc();

//Grabs camera's notification before pixel data
{ output_b(0xFA); //Debug LED
test[6] = getc(); //gets new frame = 1;
im_x = getc(); //gets x size
im_y = getc(); //gets y size
pix_sum = 0; //Used to determine if draught line reached

//Grabs pixel data & converts to grayscale.
for(j=0;j<im_y;j++)
{ for(i=0;i<im_x;i++)
    { idiv_r = div(getc(),3);
      idiv_g = div(getc(),3);
      idiv_b = div(getc(),3);
      r = idiv_r.quot;
      g = idiv_g.quot;
      b = idiv_b.quot;
      image_prev[j][i] = (r+g+b);
      output_b(image_prev[j][i]); //output image values to LED
    }
    test[7] = getc(); //gets new column
}
test[8] = getc(); //gets end frame
    test[9] = getc(); //gets colon
}
/* //Debug LEDs to let know everything is OK after first frame gather
output_b(0xAA);
delay_ms(500);
output_b(0x00);
delay_ms(500);
output_b(0xAA);
delay_ms(500); */
output_b(0);
diff = 0;
stop = 0;
//Continually grab next frame and do analysis
//until draught level is reached.
while(!stop)
{
    //Dump Next Frame
    printf("SF\r");
    for(i=0;i<4;i++)
        test[i] = getc();

    //Grabs camera's notification before pixel data
    { output_b(0xFA); //Debug LED
      test[6] = getc(); //gets new frame = 1;
      im_x = getc(); //gets x size
      im_y = getc(); //gets y size
      pix_sum = 0; //Used to determine if draught line reached

    //Grabs pixel data & converts to grayscale.
    for(j=0;j<im_y;j++)
    { for(i=0;i<im_x;i++)
        { idiv_r = div(getc(),3);
          idiv_g = div(getc(),3);
          idiv_b = div(getc(),3);
          r = idiv_r.quot;
          g = idiv_g.quot;
          b = idiv_b.quot;
          image_now[j][i] = (r+g+b);
          output_b(image_now[j][i]); //output image values to LED
        }
    }
test[7] = getc(); //gets new column 
} 
test[8] = getc(); //gets end frame 
test[9] = getc(); //gets colon 

//Does Image Analysis 
for(j=0;j<ROWS;j++) 
{ 
for(i=0;i<COLS;i++) 
{
    //Finds Magnitude of Pixel by Pixel Difference 
    if(image_now[j][i]>image_prev[j][i]) 
        diff = image_now[j][i]-image_prev[j][i]; 
    else 
        diff = image_prev[j][i] - image_now[j][i]; 

    //Sets Previous Image Pixel = Now Image Pixel 
    image_prev[j][i] = image_now[j][i]; 

    //Determines if difference is significant 
    //if significant, image_diff[j][i] = 1 
    //else image_diff[j][i] = 0. 
    if(diff>DIFF_THRESH1) 
        image_diff[j][i] = 1; 
    else 
        image_diff[j][i] = 0; 

    //Keeps running total of pixels 
    //that are considered significantly 
    //different. 
    pix_sum += image_diff[j][i]; 
} 
}
output_b(0x0F); //Debug LED 

//Checks if beer is topped off 
//by comparing the number of significant 
//difference pixels with a threshold 
//determined experimentally. 
//If the beer is topped off, then stop 
//taking more images and stop the pump. 
//Else continue with the frame grabbing 
//and image analysis. 
if(pix_sum>DIFF_SUM_THRESH) 
{ 
    output_b(0xFF); 
    stop = 1; 
} 
else 
{ 
    output_b(0x01); 
    stop = 0; 
}

//Outputs the # of significant difference pixels 
output_b(pix_sum); 
} 
}
/Turns the Pump Off
output_low(PIN_A1);

/* ****************************************** */
/* Pump Off */
/* Beer Topped Off */
/* ****************************************** */

/* ****************************************** */
/* Program Complete! */
/* ****************************************** */