Team eyeCU

Milestone 1 and 2 Goals

Nick Bertrand, Arielle Blum, Mike Mozingo, Armeen Taeb, Khashi Xiong

Milestone 1

Hardware

|  |  |
| --- | --- |
| **Module** | Daughter Board |
| **Inputs** | 8-bit parallel image data  VD, HD, DCLK  Serial data from MSP430 board |
| **Outputs** | Buffered camera data.  Serial data to MSP430 board  VD, HD |
| **Functionality** | This board will read data from the camera board, and buffer it. The buffer utilizes DCLK to read the data in. VD and HD are sent along to the Beagle Bone to be used in Image Processing algorithms. Serial data will be both sent and received using an XBee. |
| **Demonstration Plan** | Show the actual PCB daughter board populated.  Show the power rails generated, and note any errors that need to be fixed for the next revision.  Show the layout and schematics for the next revision of the board.  Probe I2C lines to demonstrate functionality and level shifting (3.3V to 2.8V) |
| Generating the required 1.5V  Generating the required 2.8V  I2C lines level shifting correctly | |

|  |  |
| --- | --- |
| **Module** | Lighting Configuration |
| **Inputs** | Ambient lighting environment for eyeCU users. |
| **Outputs** | Modified environment with proper illumination. |
| **Functionality** | Modifies the user’s environment to meet necessary lighting conditions for the gaze tracking algorithm to work as intended. |
| **Demonstration Plan**  Lighting Setup | Show the physical lamp, demonstrate the effects of this new lighting configuration with web camera. |
|  | |

|  |  |
| --- | --- |
| **Module** | Power Supply |
| **Inputs** | 6.6V from LiFePO4 battery |
| **Outputs** | Linear Regulators output 5V, 3.3V, 2.8V, and 1.5V voltage rails to hardware components. |
| **Functionality** | To provide power to Beagle Bone, Beagle Bone XBee, and Camera. |
| **Demonstration Plan**  Power circuit schematic  Power PCB layout  Bill of materials | Show schematics and PCB layout in Altium. |

|  |  |
| --- | --- |
| **Module** | Camera board |
| **Inputs** | Visual data |
| **Outputs** | 8-bit data bus with image information in RGB 5:6:5.  HD, VD, DCLK |
| **Functionality** | The camera board houses the camera and supporting hardware. The camera will output da­­ta in the configuration above. Each pulse of DCLK signifies another 8-bit parallel chunk is ready to be read. HD signifies the end of a line in the 640x480 resolution of the final images while VD signifies the end of the image. |
| **Demonstration Plan** | Show PCB in-hand and populated |
| PCB in-hand and populated | |

|  |  |
| --- | --- |
| **Module** | MSP430 Board |
| **Inputs** | Serial data from daughterboard/Beagle Bone and  Calibration data from host computer |
| **Outputs** | Cursor movement commands |
| **Functionality** | This board will be a liaison between the Beagle Bone and the host computer, effectively shuffling data from the Beagle to the host computer and back. The data will already be in its final form. This board may be extended to have a switching mechanism to turn off/on eye tracking cursor control. |
| **Demonstration Plan**  Schematics  PCB Layout  Parts List | Show and explain MSP430 Board schematics as well as PCB layout in Altium. Also show bill of materials for board. |

Firmware

|  |  |
| --- | --- |
| **Module** | Camera Controlled by Stellaris |
| **Inputs** | I2C signals |
| **Outputs** | Control of camera |
| **Functionality** | Initialize the camera over I2C |
| **Demonstration Plan** | Blink an LED on the Stellaris evaluation board to demonstrate writing to a GPIO.  Probe I2C SDA and SCL to show that I2C is working and satisfies minimum timing requirements.  Use oscilloscope to show clock generated by camera after successful initialization of the camera. |
| LED Blink  Generation of I2C signals using Logic Analyzer  SDA and SCL satisfy timing requirements  DCLK output | |

|  |  |
| --- | --- |
| **Module** | Beagle Bone |
| **Inputs** | Data from the Camera  XBee receive data |
| **Outputs** | Control flow to the camera over I2C |
| **Functionality** | The Beagle Bone firmware is in charge of running the camera, providing data to the DSP software handling communication with the XBee module and communicating with a host computer over USB and Ethernet for debugging purposes. |
| **Demonstration Plan** | Blink a LED on the daughterboard to demonstrate writing to a GPIO.  Probe I2C SDA and SCL to show that I2C is working.  Be able to communicate via serial (USB-FTD) interface. |
| LED Blink  Communications over USB-RS232  Generation of I2C signals Logic Analyzer | |

|  |  |
| --- | --- |
| **Module** | Wireless |
| **Inputs** | Wireless data generated by Beagle Bone. UART data from the MSP430. |
| **Outputs** | UART data to the MSP430 board. Wireless data to the Beagle Bone. |
| **Functionality** | The XBee wireless will have two-way communication. The Beagle Bone Xbee will be able to send cursor commands to the MSP430 XBee. And the MSP430 XBee will send algorithm parameter data the Beagle Bone XBee. |
| **Demonstration Plan** | Send a string from the MSP430 development board through UART1 and into an XBee. The XBee will transmit this string to another XBee connected to a USB XBee explorer and will transmit the string to the computer and display the string on a terminal. |
| String Sent over XBee | |

Software

|  |  |
| --- | --- |
| **Module Name:** | threshold() |
| **Inputs:** | imageData from a pre-recorded video, initial threshold (manually chosen) |
| **Outputs:** | List of points that satisfy threshold criteria |
| **Functional Description:** | Scans each pixel in the region of interest in a frame and checks to see which pixels are dark enough to belong to the pupil. This process is repeated until a region (computed with getConnectedRegions()) with an area close to a reference area is found, or until a maximum number of iterations has been reached. If the maximum number of iterations is reached and no suitable regions are detected, identify the user as blinking. |
| **Demonstration Plan:** | Color the dark pixels as red, and visually inspect the image to ensure that pixels that meet the threshold requirement have been marked. |
| Pixels that meet threshold requirements are marked red | |

|  |  |
| --- | --- |
| **Module Name:** | getConnectedRegions() |
| **Inputs:** | List of dark points identified in threshold() |
| **Outputs:** | crPointList, crSize, crCount,crBinary |
| **Functional Description:** | Uses a stack based implementation of the flood fill algorithm to identify connected regions of dark points. |
| **Demonstration Plan:** | Color each connected region that meets the size requirement a different color, and visually inspect the resulting image. |
| Connected region correctly colored.  No color if blinking | |

|  |  |
| --- | --- |
| **Module Name:** | getAspectRatio() |
| **Inputs:** | crBinary, crCount |
| **Outputs:** | Aspect ratio for each connected region in CR, index of the connected region with aspect ratio nearest to one |
| **Functional Description:** | Computes the ratio of the longest horizontal and longest vertical lengths. The connected region with the aspect ratio closest to one is identified as the pupil. |
| **Demonstration Plan:** | Print out a list of the aspect ratios computed and visually inspect an image with the connected regions in CR. |
| Aspect Ratios visually correspond to shape | |

|  |  |
| --- | --- |
| **Module Name:** | removeAberrations() |
| **Inputs:** | crPointList, crMap, crSize, Index indicating chosen region |
| **Outputs:** | Updated crPointList and crSize |
| **Functional Description:** | Computed the number of pixels in each row of the connected region and find the mean and standard deviation of the pixel counts. Remove rows that have pixel counts that fall out of a certain number of standard deviations away from the mean. Repeat the process in the vertical direction. |
| **Demonstration Plan:** | Display the image with the chosen region before and after removal of aberrations and verify that aberrations have indeed been removed. |
| Aberrations are removed in satisfactory manner. | |

|  |  |
| --- | --- |
| **Module Name:** | computeCentroid() |
| **Inputs:** | crPointList, crSize, Index indicating chosen region |
| **Outputs:** | Coordinates of the centroid |
| **Functional Description:** | Sum the coordinates of all points belonged to the pupil region and divide by the total number of points. The result is the coordinate of the centroid. |
| **Demonstration Plan:** | Indicate the centroid with horizontal and vertical lines and verify by visual inspection that the intersection falls on the centroid of the region. |
| Centroid matches with visual inspection | |

|  |  |
| --- | --- |
| **Module Name:** | generateCursorCommand() |
| **Inputs:** | Reference centroid coordinates, current centroid coordinates |
| **Outputs:** | Cursor command code |
| **Functional Description:** | Compares the reference centroid coordinates to the current centroid coordinates. If the difference between the two coordinates exceeds a threshold value for a fixed number of consecutive frames, then the cursor command output will be changed accordingly. Otherwise, the previous cursor command is output. |
| **Demonstration Plan:** | We will manually pick a frame from which to configure the reference centroid. The input video will have a known sequence of gestures (for example, the user will look up, blink, look left, blink, etc.) The cursor command generated will be output in real time on the console as the video is processed. |
| Cursor Command matches the direction of gaze | |

Milestone 2

Hardware

|  |  |
| --- | --- |
| **Module** | Daughter Board |
| **Inputs** | 8-bit parallel image data  VD, HD, DCLK  Serial data from MSP430 board |
| **Outputs** | Buffered camera data.  Serial data to MSP430 board  VD, HD |
| **Functionality** | This board will read data from the camera board, and buffer it. The buffer utilizes DCLK to read the data in. VD and HD are sent along to the Beagle Bone to be used in Image Processing algorithms. Serial data will be both sent and received using an XBee. |
| **Demonstration Plan** | Have the 2nd revision of the daughterboard populated and show functionality of the components by probing the power rails, I2C lines, camera data and the ability to read data in software. |
| Show latest PCB populated | |

|  |  |
| --- | --- |
| **Module** | Power Supply |
| **Inputs** | 6.6V from LiFePO4 battery |
| **Outputs** | Linear Regulators output 5V, 3.3V, 2.8V, and 1.5V voltage rails to hardware components. |
| **Functionality** | To provide power to Beagle Bone, Beagle Bone XBee, and Camera. |
| **Demonstration Plan** | Measure all voltages of power PCB revision one. All values must be within 5% of their designed value. |
| 5V ± 5% | |

|  |  |
| --- | --- |
| **Module** | Camera board |
| **Inputs** | Visual data |
| **Outputs** | 8-bit data bus with image information in RGB 5:6:5.  HD, VD, DCLK |
| **Functionality** | The camera board houses the camera and supporting hardware. The camera will output da­­ta in the configuration above. Each pulse of DCLK signifies another 8-bit parallel chunk is ready to be read. HD signifies the end of a line in the 640x480 resolution of the final images while VD signifies the end of the image. |
| **Demonstration Plan** | Show most recent PCB revision including physical board, schematics and PCB layout in Altium. Show the correct voltages generated by the 1.5v and 2.8v linear regulators, show the DCLK line, and probe data lines to show level shifting. |
| Show latest PCB  Verify functionality (verified in firmware)  1.5V generation  2.8V generation  Level shifting for I2C  Level shifting for camera data | |

|  |  |
| --- | --- |
| **Module** | MSP430 Board |
| **Inputs** | Serial data from daughterboard/Beagle Bone and  Calibration data from host computer |
| **Outputs** | Cursor movement commands |
| **Functionality** | This board will be a liaison between the Beagle Bone and the host computer, effectively shuffling data from the Beagle to the host computer and back. The data will already be in its final form. This board may be extended to have a switching mechanism to turn off/on eye tracking cursor control. |
| **Demonstration Plan** | Have PCB in-hand and populated. Probe 3.3V power to verify within 10% tolerance. Show functionality of JTAG programming the MSP430. Show list of changes necessary for next revision. |
| PCB populated and in-hand  MSP430 JTAG Programming  Tentative PCB revisions  XBee functional | |
|  | |

|  |  |
| --- | --- |
| **Module** | eyeCU Glasses |
| **Inputs** | Camera board and periphery |
| **Outputs** | Structural support for the camera board. |
| **Functionality** | To provide physical interface between the eye tracking system and the user |
| **Demonstration Plan** | Show glasses, demonstrate the mounting structure of the camera on the glasses. Demonstrate a user wearing the device. |
| Glasses  Mounting structure  Demonstrate use | |
|  | |

Firmware

|  |  |
| --- | --- |
| **Module** | Beagle Bone XBee |
| **Inputs** | 1.) Output from the module generateCursorCommand() is the input to the Beagle Bone XBee.  2.) Algorithm parameters from Host Computer Calibration module. |
| **Outputs** | Wireless data transmission |
| **Functionality** | The Beagle Bone will send the data output by the generateCursorCommand() module wirelessly. |
| **Demonstration Plan** | Send data wirelessly from the Beagle Bone XBee to an XBee connected to a computer with an XBee USB Explorer. |
|  | |

|  |  |
| --- | --- |
| **Module** | 2-Way Wireless Data Transmission (Integration) |
| **Inputs** | Wireless data generated by Beagle Bone. UART data from the MSP430. |
| **Outputs** | UART data to the MSP430 board. Wireless data to the Beagle Bone. |
| **Functionality** | The XBee wireless will have two-way communication. The Beagle Bone XBee will be able to send cursor commands to the MSP430 XBee and the MSP430 XBee will send algorithm parameter data the Beagle Bone XBee. |
| **Demonstration Plan** | Send data wirelessly from MSP430 XBee to Beagle Bone XBee. Send data wirelessly from the Beagle Bone to the MSP430. |
| Data sent from MSP430 XBee to Beagle Bone XBee  Data sent from Beagle Bone XBee to MSP430 XBee | |

|  |  |
| --- | --- |
| **Module** | Beagle Bone |
| **Inputs** | Data from the Camera  XBee receive data |
| **Outputs** | I2C Camera control signals |
| **Functionality** | The Beagle Bone firmware is in charge of running the camera, providing data to the DSP software handling communication with the XBee module and communicating with a host computer over USB and Ethernet for debugging purposes. |
| **Demonstration Plan** | Demonstrate that the firmware can initialize over I2C.  Show the ability to communicate over USB 2.0 or Ethernet.  Show the ability to be able to save images to the SD card. |
| USB 2.0 communications (used for streaming video, however demonstrate any communication)  Storage of data on SD card | |

Software

|  |  |
| --- | --- |
| **Module** | Serial Communication (Beagle Bone and host computer) |
| **Inputs** | 640x480 RGB calibration frames from serial communication module |
| **Outputs** | Queue of video frames |
| **Functionality** | Packets of video data are received via USB from the Beagle Bone. Once an entire frame of data has been received, the frame is stored in an image data array, and a pointer to the array is pushed onto a video frame queue.  Note: video is only transmitted from the Beagle Bone during the calibration process. Once calibrated, we will no longer need to display the video data sent from the Beagle Bone. |
| **Demonstration Plan** | Send simulated data from the Beagle Bone via USB and demonstrate that it is correctly received by displaying it on host computer. |
| Demonstrate data was received | |

|  |  |
| --- | --- |
| **Module** | Serial Communication (MSP430 and host computer) |
| **Inputs** | Packets from MSP430 over USB |
| **Outputs** | Data extracted from packets |
| **Functionality** | Host Computer receives packets from the MSP430 at rate of 30Hz, extracts commands codes from the packet and puts them into a queue to wait to be processed by the cursor movement module. |
| **Demonstration Plan** | Expect the MSP430 breakout board to output a set of simulated data and verify that it is correctly received by the host computer. On the Host Computer-side the data is inspected to verify that is the same as the data that was sent from MSP430.  (Display data on screen) |
| Show data was correctly received | |

|  |  |
| --- | --- |
| **Module** | Calibration |
| **Inputs** | 640x480 RGB calibration frames data sent via USB from the Beagle Bone |
| **Outputs** | Reference pupil centroid, reference pupil area, processing region |
| **Functionality** | A GUI interface on the host computer displays the images being captured by the camera with an overlay of the processing. The GUI allows the user to modify algorithm parameters. After some parameters are chosen manually, the user will be guided through a process to collect the remaining calibration values. |
| **Demonstration Plan** | Visually verify that the modified parameters result to a modified overlay. |
| Visually verify modified parameters change overlay | |

|  |  |
| --- | --- |
| **Module** | GUI |
| **Inputs** | User adjusted parameters |
| **Outputs** | Visualization of the image processing |
| **Functionality** | Allows the user to control parameters in the algorithm such as initial threshold value and cursor speed. The image processing is visualized by overlaying colored regions over the original image. Also it allows user to pause eye controlled cursor movement. |
| **Demonstration Plan** | Test to see if changes in parameters correspond to correct modification in the overlaying image. |
| Show user interface  Show user has control over parameters | |

|  |  |
| --- | --- |
| **Module** | Indicators |
| **Inputs** | User starts calibration |
| **Outputs** | Visual cues to guide the user through calibration, Reference pupil centroid, reference pupil area, processing region |
| **Functionality** | Displays on-screen indicators to tell the user to look at a series of calibration points. From these points, a processing region and reference pupil size and location can be determined. |
| **Demonstration Plan** | Save the frames used for each step in calibration, and visually verify that the parameters generated are correct. |
| Demonstrate the visual cues  Parameters are generated correctly by visual inspection | |