CSC 453 - Homework 2

Wolfpack J

Jason Benckert, Cameron Finley, Stanton Parham, Weirui Wang

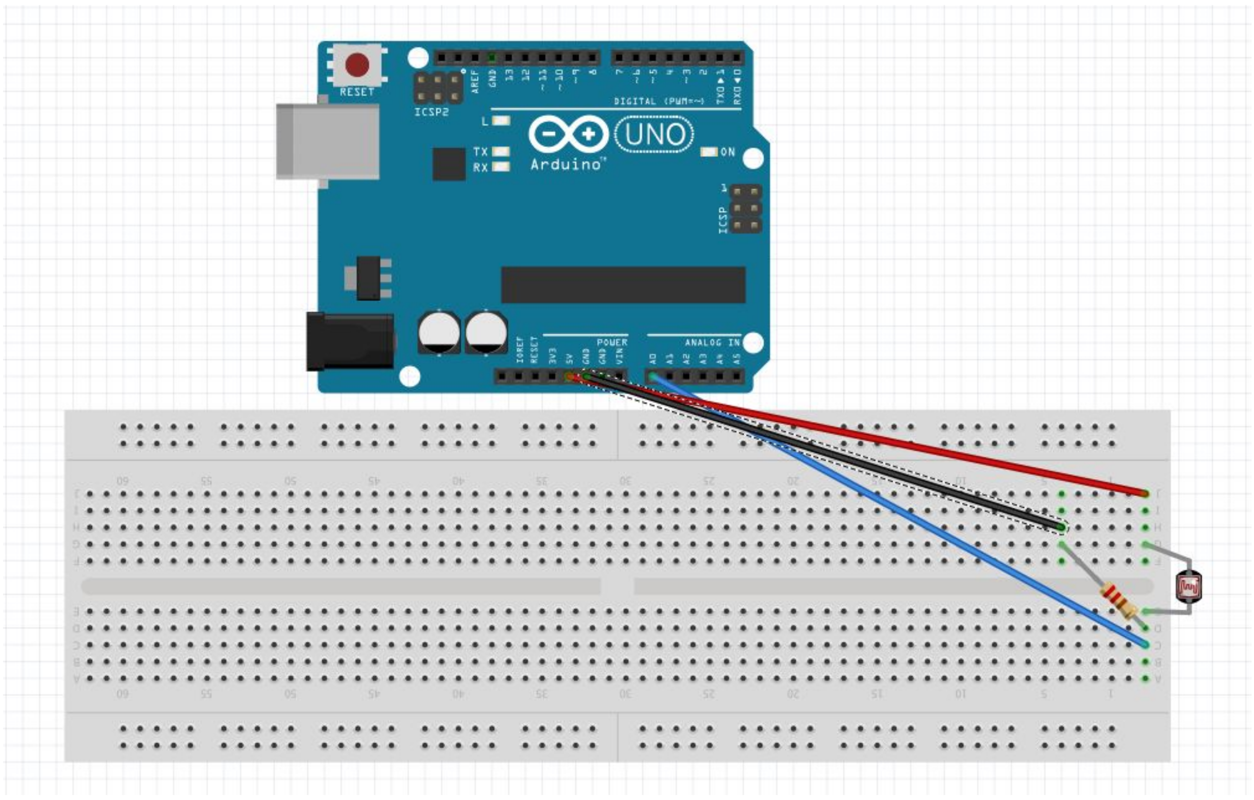
Percentage of Contribution

Jason Benckert - 25%
Cameron Finley - 25%
Stanton Parham - 25%
Weirui Wang - 25%

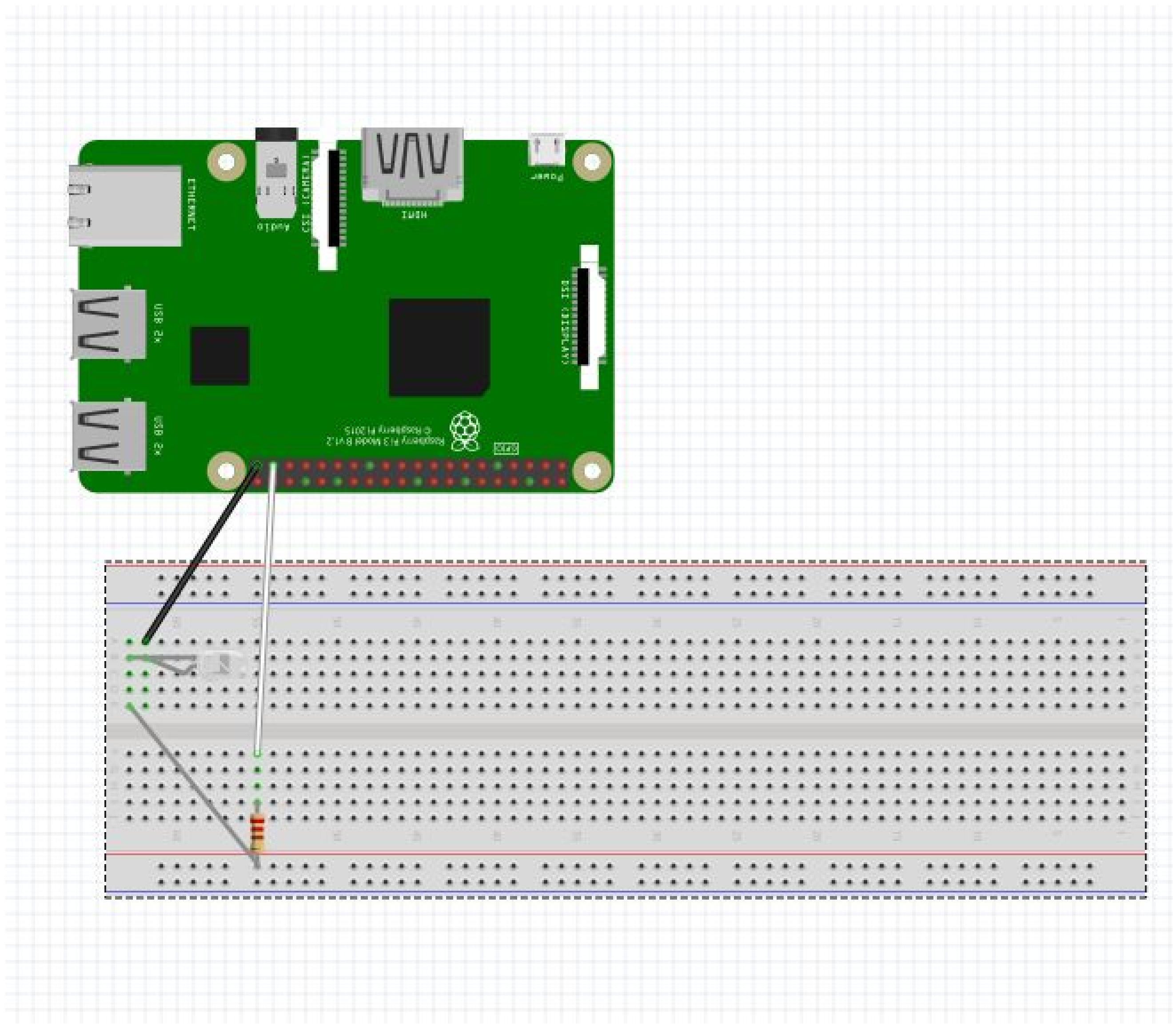
Subtasks

Task	**Jason Benckert**	**Cameron Finley**	**Stanton Parham**	**Weirui Wang**
setting up Laptop #1	30%		70%	
setting up Raspberry Pi		20%	60%	20%
setting up Arduino		25%	30%	45%
setting up Laptop #2	100%			
testing and fixing the entire IoT system	25%	25%	25%	25%

Arduino Schematic



Raspberry Pi A Schematic



Design

Header and packet from Raspberry Pi to Arduino

When creating the header and packet to send between the Raspberry and Arduino we needed a way to send the necessary data. The header was important in informing the Raspberry on how laptop 2 was suppose print out the output. When creating the header we decided it should be 16 bits in order to best manage the data so that we would provide as much of the payload that is possible without sacrificing the space needed for the header. The first 8 bits feature the message id. This ID helps to identify the state of the data being sent to the Arduino. When the message id is part of the last batch of data sent it is returned to laptop 1 to confirm that all of the data was sent without issue.

The last 8 bits contain important parameters needed to determine how the output is printed. Bit 1 determines what type of output is done. This makes it possible for the laptop 2 to know whether to print the output or write it to a file. The second bit determines whether to payload is the end of a file. This is key for transferring files letting laptop 2 know that all of the file is transferred so that the program can properly close the file.The final 5 bits determine the message size. This makes it possible for Arduino know how much data to send to the laptop so that no data is lost or wasted.

laptop#2 to Raspberry Pi

It is important for laptop 2 to be able to communicate with the Raspberry to inform when the data was completely sent. To accomplish this we had it so that the ack was made up of an 8 bit byte provided from the message id of the message header. This helped to provide a status of the transfer where once it was done, the message id would be a transfer completed. Once the cycle is done the ack with the complete message would be sent to the raspberry to inform it of the success of the transfer. To take into account the chance of an incomplete data transfer, the ack would be assigned none before the start of a cycle so that if a transfer is incomplete the ack will be sent as none letting the Raspberry know that the transfer was a failure.

LED transmit a 0 or a 1

For transmitting a 0 or a 1 through the led, we went with a slow yet simple way to transmit the data. After a simple synchronizing process, the raspberry Pi would go through a cycle. During the cycle the Pi will scan the next bit in a message the program created from the data received from laptop one every ten milliseconds. For each bit the Pi will either turn on or turn off the light depending on the next bit the program read. If the bit is 1 the program with turn on the light, or nothing if the light is already on. If the bit is 0 the program turns off the light, or doesn’t do anything if the light is already off. This continues until the program goes through the entire message.

While this is going on, the arduino is going through a cycle parallel to the raspberry Pi, After going through a synchronizing process,the Arduino checks for possible light changes. Every ten milliseconds the Arduino checks to see if the light on the Raspberry is on. If it is the Arduino rights a 1 in a variable, if not then it writes a 0 instead. During the process, if the variable reaches a length of eight, it is converted to a binary object which is written to the Serial to be sent to laptop 2. This continues until the Raspberry’s light is off for an extended period of time showing that the Raspberry is done transmitting data.

Synchronizing Raspberry Pi with Arduino

It is important for the Raspberry Pi and Arduino to be in sync so that no data is lost during the transmission of data. To accomplish this we first set a special procedure where the Raspberry sends signal to the Arduino letting it know that it is about to send data. This is accomplished by having the Raspberry turn on its light for five milliseconds before it starts the process of flickering the light on and off to send binary data.

On the arduino side, it has a wait boolean state. This allows the Arduino know when to start its cycle to write data. As the program runs it keeps an eye on whether or not the light is on. When the light turns on the Arduino keeps track of how long the light is on for. Keeping track of the light time informs on whether or not the raspberry is wanting to send data. If the light has been on for at least five milliseconds this causes the Arduino to change its wait boolean letting it know to start its cycle. During the cycle, the program also has a separate set of variables that keep track for how long a light is off during the cycle. This is done so that if the light has been off for a certain amount of time this would mean the raspberry pi is done transmitting data. This condition switches back the wait boolean so that the Arduino doesn't send any false data to laptop 2.

Transmit Length

In order to try to prevent the risk of lost data during the transfer process, we had it so that data was transmitted one bit at a time. Because both the Raspberry and Arduino were designed to receive/send a bit every ten milliseconds, this would mean that the Arduino would receive 100 bits a second leading to at least 13 bytes of data a second. This would lead to a long time for data to transfer between the Raspberry and Arduino. For the 200 byte file it would take at least.15 seconds. The 1kb file would take one and a half minutes. Finally, the 10kb file would take about 13 minutes. While this protocol is done as a way to lower the risk of data lost, it clearly isn't the best in terms of data transfer speed and wouldn't be good for more complex systems.

Scheme to build reliability

While are design has slow data transfer, we strived to try to make sure the data flow went without problems. To make this possible we made sure to set all messages, packets, and arks in even 8 bit bytes. Doing this made sure that the data was easy to transfer between machines but also make sure that there was no wasted data. We also worked hard to make necessary configurations in terms of timing so that our Raspberry and Arduino were synchronized during data transfer. These configurations made it so that all data transfer would make it to its necessary destination with no lost in data. In order to manage when Arduino waits and when it writes output, we made sure to incorporate several variables to that it can know when it is suppose to write as well as keep track of timing so that it knows when to stop as well. All these factors help to insure that no data is lost during the transfer.