Chris Bird, Colin Lee

EEC 172: Embedded Systems

Due: 27th February, 2015

TivaWare – Xbee Module Integration

Using XBee modules, we were to implement three different tasks on our processors. First, we had to do a simple loopback test using a single processor, local XBee, and a remote XBee that had its DOUT and DIN wired together. With this setup we could send a message to the remote XBee and immediately get it back to our local XBee. The modules were put into transparent mode, with our given address and PAN ID. This allowed us to send and receive messages without having to add on the extra protocols such as checksum. After this, we tied the remote XBee to the other processor and started doing sending and receiving between processors. With this we were able to complete the second task, which was wireless texting. Our lab two code was sufficient for this, except we needed to add a little extra to ensure that all data received was both correct, and that correct data was routed to the correct place.

Finally, after this was working and correct, we coded a game of pong, and started troubleshooting. This is where our main difficulty was, as not only was our games not synching up properly, but when transmitting multiple pieces of data we needed to ensure that they were used properly. We eventually deciding on sending a “start” bit of sorts, when a processor received a certain character, it would set ball coordinates and update the ball, and when it received a different character, the next items would be paddle coordinates. In our final implementation, our paddles were moving correctly, and synching properly onto the other, along with the ball. We achieved this by having player one control the ball movement, and constantly sending updated coordinates. Then both player one and player two controlled their own paddle and send coordinates for the top pixel to the other processor, which updated the position of the paddle.

The only difference between our code and what a viewer might expect, is that on each screen, the player using that screen is always on the left side regardless of which player. On screen one, with player one, he was the blue paddle on the left, and player two was the red paddle on the right. On player two’s screen (screen two) he was the blue paddle on the left, with the red paddle on the right being player one. We achieved this by inverting the ball coordinates before sending to the other player, along with printing received paddle positions on the right side regardless of player.

No significant difficulties were encountered in this lab. What we spent most time on was getting the XBee to correctly connect to each other in a way that would send data quickly. We determined that we cannot set internal settings for the remote XBee while the local one is in transparent mode, which is interesting to note, but very easily understood as to why this is the case.

Code Description and Walkthrough

Our code includes all of the OLED tests and initializations. This is due to the fact that we copied our lab two code over in order to create this, as well as running through some or all of the tests on each run in order to show that the screen was turning on properly. We now only run the LCDTestLines function on each start, it helps with debugging hardware issues and resembles the standard loading screen of many programs. For our handlers we have the handlers for receiving standard input from the IR Receiver which worked in previous codes; along with this we have one for monitoring the Rx pin to connect with UART1. Once we receive anything on this pin, we interrupt, read all data being sent, then return to the program. This takes a very short time to complete.

Aside from the handlers, our main code is the DID contain the texting code, which was later removed to make our Pong game more readable. Our player one code, which is clearly marked, at the beginning of the code, has a function called “UpdateBallP1” which controls the motion, slope, and coordinates of the ball. It also controls collisions, and checks for paddles, side walls and upper walls. If a player wins, it calls the reset game function, which returns the ball coordinates to the center of the screen and resets. If it hits the paddle or the top/bottom walls, it updates the ball slope and redraws.

The only difference between player one code and player two code is that player once calls the update ball function and reset game function while player two does not. Along with this, player two has an extra part in his interrupt handler that when he receives a message over the XBee he checks whether it is the paddle or the ball before acting on the data received, whereas player one only works with paddle coordinates from player two.

There are some artifacts and comments left over from the lab two code, along with some leftover or unused parts. These can be safely ignored as they do not run in the final code.