Lab 4

Kyle Jeffrey

Abstract—The fourth lab uses the PSoC Micro and the Raspberry PI and UARTS for communication between the two.

1 Introduction

This lab introduces the Development on the Raspberry Pi. The development environment I used was Visual Studio Code with an SSH addon. The UART Communication protocol is used in both a software and hardware implementation and the PSoC 6 will be communicating with the Raspberry Pi using the hardware and software version of the UART protocol.

2 Part 1: Remote Control of LED brightness with PWM

In this part, the brightness of an LED is controlled with the Raspberry Pi using PWM. A potentiometer is connected to the PSoC 6 using an ADC which then transmits the data over UART to the Raspberry Pi, changing the PWM signal.

2.1 PSoC

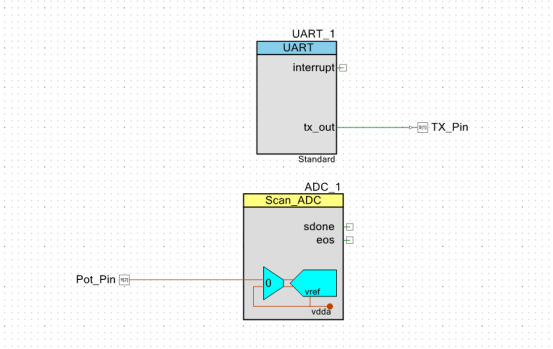

Fig. 1. Hardware Setup

The PSoC polls sends the ADC Data in the main loop through the UART with a 1ms delay. No interrupts were used.

2.2 Raspberry Pi

The Pi uses the Wiring Pi Library to enable using the GPIO Pins. For this part, it is just configured to read in data.

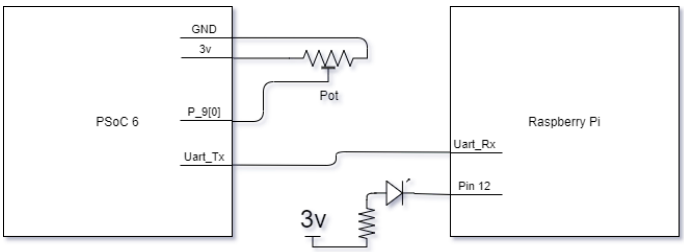

Fig. 2. Top Level

2.3 Testing

This section went pretty smoothly. To assure that the UART was working on either end though, the AD2 oscilliscope was used to check that the PSoC was properly transmitting data.

3 Part 2: Analog Loopback through the Raspberry Pi

For this part, the ping pong buffer system used in lab 2 was repurposed. An ADC using DMA filled two buffers in a ping pong buffer style system. When one buffer was ready, it was transferred over UART to the PI and the PI would then send the Data back over UART to the PSoC which then turned the ADC data back into an analog output. A sinewave was feed at the ADC Input, and a similar sinewave output was expected at the output.

3.1 PSoC

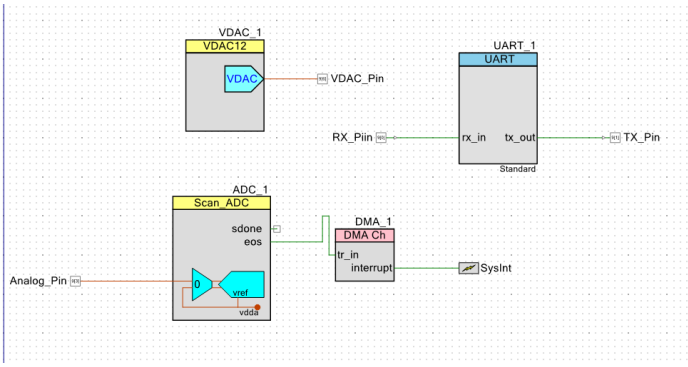

Fig. 3. Hardware Setup

Notice that most everything is occuring on the PSoC side of the design. The Pi simply recieves the data over UART and then returns it.

3.2 Raspberry Pi

The Pi design was fairly simple, and used the software implementation of UART. It had to do an extra step in this instance to assure that all of the data recieved was sent. A check was done after every send to make sure of this.

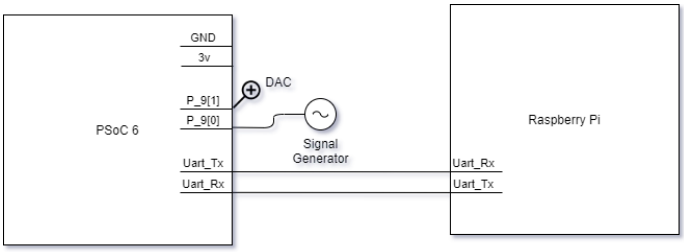

Fig. 4. Hardware Setup

3.3 Testing

It was essential to eliminate all possible sources of error in this part because there was a lot of moving parts on the design. The UART's on the PSoC and Pi side were both tested to assure proper data transmission out of the TX pin. The data was also increased incrementally in size beginning with sending hello worlds over the UART from the PSoC.

3.4 Comments

The sinewave was able to maintain stability up to 100hz but higher than that, the signal would begin to grow lower resolution, and noticeables steps could be seen in the waveforms. Faster Polling on the ADC could help as well as using an ISR for the recieve AND send UART on the PSoC.

4 Conclusion

This lab was an introduction into the Raspberry Pi Development environment as well as UART communication between multiple devices. It also introduced the subtlties in resolution scaling, as even though the ADC had a 12 bit resolution, UART only supports an 8 bit payload. There are an infinite number of uses for UART, at is a fairly cheap, but decently fast communication protocol. This setup with the PSoC and Pi could be used for tasks that require a fast general purpose CPU with a highly configurable Microcontroller like the PSoC.

Kyle Jeffrey is a Senior Robotics Engineering Student at the University of California Santa Cruz. He is the Secretary of the Engineering Fraternity Tau Beta Pi and the lead Hardware Engineer at the on campus startup Yektasonics.