Microcontroller Block

ECE441 Wearable Sensor for the Blind Project

Samuel Jia Khai Lee

ECE Senior Design

Oregon State University,

Corvallis, OR USA

leejiaks@oregonstate.edu

Keywords—Microcontroller; block description; design verification

# Introduction

The purpose of this document is to describe the Bluetooth block of the ECE441 Wearable Sensor for the Blind project to 4th year ECE students with the intention of these students being able to build and verify the block without further research. The Microcontroller block is implemented using Arduino Pro Mini 5V module. This document provides an overview of the overall block function including interface properties and a schematic, verification for the design in the form of a step-by-step testing process, and support for the validity of the design in the form of outside research and numerical justification addressing individual properties.

# Design Details

The wiring diagram (Fig.1) presents the block design, including the interfaces of the block. These interfaces are further elaborated through validation information given in Table 1, which demonstrates externally-supported validly for each of the block’s properties.

GRN RX0 RXI VCC GND BLK

RAW

GND

RST

Vcc

A3

A2

A1

A0

13

12

11

10

pwr\_spply\_mcrcntrllr\_dcpwr

mcrcntrllr\_cntrllr\_cd\_data

Tx0

RXI

RST

GND

2

3

4

5

6

7

8

9

pwr\_spply\_mcrcntrllr\_dcpwr

Microcontroller

mcrcntrllr\_cntrllr\_cd\_data

1. Black Box Diagram of Bluetooth Block

# Block Overview

The Microcontroller contains data and instructions for the system controller. The block is powered from **pwr\_spply\_mcrcntrllr\_dcpwr** interface usually around 5V with a nominal current draw of 50 mA. Fig. 1 shows the black box diagram. Data are sent via a serial port protocol over the **cntrllr\_cd\_bltth\_data** interface. Finally, the output is represented by the **bltth\_cntrllr\_cd\_rf** interface. The function of the Microcontroller block is storing and processing data, and running the I2C interfaces of the system. This block is completed by Samuel Lee.  
  
A full listing of interface properties can be found in Table 1.

mcrcntrllr\_cntrllr\_cd\_data

Pwr\_spply\_mcrnctrllr\_dcpwr

Microcontroller

1. Black Box Diagram of Bluetooth Block

`

| Interface | Properties |
| --- | --- |
| **pwr\_spply\_mcrcntrllr\_dcpwr** | 1.Min Voltage: 1.8V  2.Max Voltage: 5.5V  3.Nominal Current: 9mA  4.Peak Current: 20mA |
| **mcrcntrllr\_cntrllr\_cd\_data** | 1.Total Code Size: 10KB  2.Language: Arduino  3.Clock Frequency: 20MHz |

1. Microcontroller Block Interfaces and Properties

# Block Verification

Based on the interfaces for this block, a verification (testing) process needs to be indicated. This will allow the final constructed design to be tested verifying that all the interface properties have been met and that the block is ready for integration into the system.

## Power Testing (**pwr\_spply\_mcrcntrllr\_dcpwr**)

This test will verify that the Microcontroller operates after providing **pwr\_spply\_mcrcntrllr\_dcpwr** interface.

1. Connect the Microcontroller block to power via the **pwr\_spply\_ mcrcntrllr\_dcpwr** interface at RAW pin.
2. Connect a multimeter to the **pwr\_spply\_ mcrcntrllr\_dcpwr** to check the voltage.
3. Load a test program (Arduino->Files->Examples->Basics->Blink) to the system microcontroller.
4. Modify the delay of the code and load it into the system microcontroller.

PASS: This test passes if 9 out of 10 individuals are able to successfully power up the Microcontroller by illuminating and modifying the blinking intensity of the LED on the Microcontroller.

[Link to Video](https://drive.google.com/open?id=1cwFrvuvX5zWKP3GXsvQ9lN-6Gond0Vwy)

## Microcontroller Testing (**mcrcntrllr\_cntrllr\_cd\_data**)

This test will verify the **mcrcntrllr\_cntrllr\_cd\_data**.

1. Step 1.

[Link to Video](https://drive.google.com/open?id=1cwFrvuvX5zWKP3GXsvQ9lN-6Gond0Vwy)

If the block passes all the listed tests, all interface properties have been verified and the block is ready for inclusion into the system.

# Design Validation

For this block, an off the shelf solution was chosen. The ATmega328P on the Arduino Pro Mini microcontroller fits the needs of the block interfaces without needing to be modified. Table 2 includes the interface property validation for this block. All the interface properties have been addressed and the design meets or exceeds the properties.

| Interface | Properties |
| --- | --- |
| **pwr\_spply\_mcrcntrllr\_dcpwr** |  |
| Min Voltage: 1.8V | ATmega328P on the Arduino Pro Mini Microcontroller has an operating voltage range of 1.8 – 5.5V. There is a voltage regulator on board, so it can accept voltage up to 12V [2]. (Section 28 ATmega328 Datasheet, Page 313[1]) |
| Max Voltage: 5V | ATmega328P on the Arduino Pro Mini Microcontroller has an operating voltage range of 1.8 – 5.5V. There is a voltage regulator on board, so it can accept voltage up to 12V [2]. (Section 28 ATmega328 Datasheet, Page 313[1]) |
| Nominal Current: 9mA |  |
| Peak Current: 20mA |  |
| **mcrcntrllr\_cntrllr\_cd\_data** |  |
| Total Code Size: | ATmega328P has 32K Bytes of In-System Self-Programmable Flash program memory. (ATmega328 Datasheet Page 1 [1]) |
| Language: Arduino | Controller code is programmed by using Arduino IDE to interact with the Bluetooth module. (Arduino [3]) |
| Clock Frequency: 20MHz |  |
|  |  |

1. Interface Property Validation for the Microcontroller Block

# Bills of Materials

Table 3 lists the bills of materials used for the Bluetooth block.

| Item | Price | Link |
| --- | --- | --- |
| Arduino Pro Mini 5V | $9.95 | [Link](https://www.digikey.com/product-detail/en/sparkfun-electronics/DEV-11113/1568-1055-ND/5140820) |

1. Bills of Materials for the Bluetooth Block

[1] Atmega328 Datasheet, <<http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf>>

[2] Arduino Pro Mini, <<https://store.arduino.cc/usa/arduino-pro-mini>>

[3] Getting Started with the Arduino Pro Mini, <<https://www.arduino.cc/en/Guide/ArduinoProMini#toc2>>