California State University, Fresno

Electrical and Computer Engineering Department

Experiment Title: Project Proposal

Course: ECE 186A: Senior Design I

Instructor: Dr. Raeisi

Prepared by: Nick Morley
Jonathan Richards

Date _____ March 9, 2016

To be filled out by the instructor

Comments: ______________________________

____

____

____

Final Grade:

TABLE OF CONTENTS

1 THE TEAM

Team Name:	Sensor Network
Team Members:	Nick Morley Jonathan Richards
Technical Advisor:	Dr. Hovannes Kulhandjian
Course Instructor:	Dr. Reza Raeisi

2 GOALS

- Create a system to collect, transport, analyze, store, and visualize sensor data
- Control external systems manually or automatically according to sensor inputs
- Access user interface anywhere with an internet connection

3 OBJECTIVES

3.1 Modularity

- The network shall be modular and be composed of a wireless network of similar nodes with one base station
- The nodes shall be modular and support a variety of sensors and controllers interfacing over a common hardware/software interface

3.2 Dynamic

- The network shall be able to respond to the dynamic addition or removal of nodes
- The nodes shall support hot-plugging of sensors
- The client application shall view live data from the sensors

3.3 Interactive

- The nodes shall generate alerts for the user to respond to
- The user shall control the node's sensors and controllers from the client application

3.4 Easy to Use

- The user application shall allow easy configuration of new nodes and sensors
- The historic data shall be presented in a clean, graphical manner to the user
- The user interface shall be intuitive and user friendly

4 BACKGROUND

4.1 Microcontroller Programming and Interfacing

Both team members of our team have experience programming a wide variety of microcontrollers to interface with the outside world. This knowledge will be vital to interfacing with the sensors to gather data into our network and controlling exterior devices. It will also be useful for interprocessor communication to relay messages from one end of the network to the other (including the user's phone).

4.2 Wireless Communication

Nick has some experience with establishing communication between a few processors through multiple point to point links. This will help us get starting, but we will put further research into how to implement a mesh network, including routing algorithms.

4.3 Data Storage, Processing, and Serving

Both team members have some server-side experience in spinning up servers and presenting data to the user from a database. Nick also has experience funneling data into the database that will be important for establishing communication between the server and base station to transfer data into the database. Since our sensor data will be highly compressable, we will implement a custom compression algorithm to minimize network and server resources.

4.4 Mobile App Development

Both team members have some web application experience we will leverage for first creating a web interface to our network. We will then port our web application to Android and iOS.

5 FEASIBILITY

5.1 ESP8266 Microcontroller

The ESP8266 microcontroller is a low cost microcontroller with integrated, long-range wireless communications that will be the master control unit of each node. The most important features for our purposes are:

- Unit cost of $2.44 in lots of 10 or more
- Range of 366 meters with standard consumer wireless router and range of 3.71 kilometer with long range access point
- I^2C interface for communicating with sensor modules
- Analog to digital converter for monitoring supply voltage (if we deploy battery powered nodes)
- UART port (if we interface multiple ESP8266s on a node)
- 13 general purpose input/output pins for buttons and status LEDs

5.2 ATmega328P Microcontroller

The ATmega328P microcontroller is a low cost, general purpose microcontroller. To simplify the software and hardware interface between the sensors and our network, each sensor module will contain an ATmega328P. This enables a simple, common interface between the node and sensor module over I^2C and offloads the sensor monitoring from the ESP8266. It also makes the sensor modules more universal since adding or changing sensors will not require a firmware update on the node itself. Instead the sensor specific firmware is stored on the ATmega328P that is bundled with the sensor. The most important features for our purposes are:

- Unit cost of $1.21 in lots of 10 or more
- Wide variety of intefaces to talk to the sensor based on it's specific needs
  - 20 genneral purpose input/output pins
  - 8 channel analog to digital converter
  - 6 pulse width modulation outputs
  - SPI
  - UART
  - I^2C
  - OneWire

5.3 2.8" TFT LCD Touchscreen

The touchscreen allows the node to have a direct and convienent user interface. For example, on a thermostat it would allow the user to see the current temperature, change the set point, view the energy usage of the air conditioner, and view a plot of the temperature without having to pull up the client app. It can also be used to aid in the initial setup process. It is not strictly necessary, but offers a polished feel to the system for the user to immediately see the system working. The most important features for our purposes are:

- Unit cost of $8.19
- Full color screen for displaying user interface
- Touch enabled for user input

5.4 DHT22 Sensor

The DHT22 temperature and humidity sensor can serve for either a weather station or thermostat node. It interfaces with the ATmega328P over a digital bus. The most important features for our purposes are:

- Unit cost of $5.99
- Temperature range of -40 - 80 °C
- Temperature accuracy of ± 0.5 °C
- Humidity range of 0 - 100% RH
- Humidity accuracy of ± 2% RH

5.5 AC Current Sensor

The current sensor is non-invasive and plugs into a 3.5 mm TRS connector. Since it is non-invasive, the client safely clips it around the device they would like to measure without having to cut any wires. The most important features for our purposes are:

- Unit cost of $11.19
- Measures up to 100 A loads (can be used for air conditioners or entire house)
- Common connector

5.6 Barometric Pressure Sensor

The barometeric pressure sensor allows us to enhance a weather station's measurements or track the node's elevation. The most important features for out purposes are:

- Unit cost of $4.34
- Pressure range of 300 - 1100 hPa
- Pressure accuracy of 0.02 hPa
- Elevation range of -500 - 9000 meters
- Elevation accuracy of 17 cm
- I^2C interface

5.7 Brightness Sensor

The brightness sensor is a photoresistor that can detect the ambient brightness level of the room. It can be used to monitor when people leave the lights on or track sunrise/sunset. We can also mount multiple sensors on each module pointed in different directions The most important features for our purposes are:

- Unit cost of $0.03 in lots of 100 or more

5.8 Audio Sensor

The audio sensor is a microphone that can detect people talking or serve as a spectrum analyzer using digital signal processing.

5.9 Cloud Server

The cloud server will store historic sensor data and serve the web application for the users in addition to hosting the project management software for this project. It is the single entry point for users to access the sensor network from the internet.

6 PROTOCOLS

6.1 Intranode Protocol

The Intranode Protocol is used to gather sensor data from the sensor and controller modules to the node. The physical layer diagramed in Figure 6.1 depicts pins connecting the systems. The link layer is composed of 32 byte packets exchanged between the systems as described in Table 6.1. All words are stored in little-endian format.

Table 6.1: Intranode Packets

Field Name	Byte Index	Field Width	Field Explation
Address	0	8 bits	Which port the sensor is plugged into
Message Type	1:2	16 bits	An identifier for how the payload is formated
Sensor Type	3:4	16 bits	An identifier for the physical sensor board
Reserved	5:6	16 bits	Reserved for later revisions
Payload	7:30	24 bytes	Message data
Checksum	31	8 bits	Checksum of first 31 bytes of packet

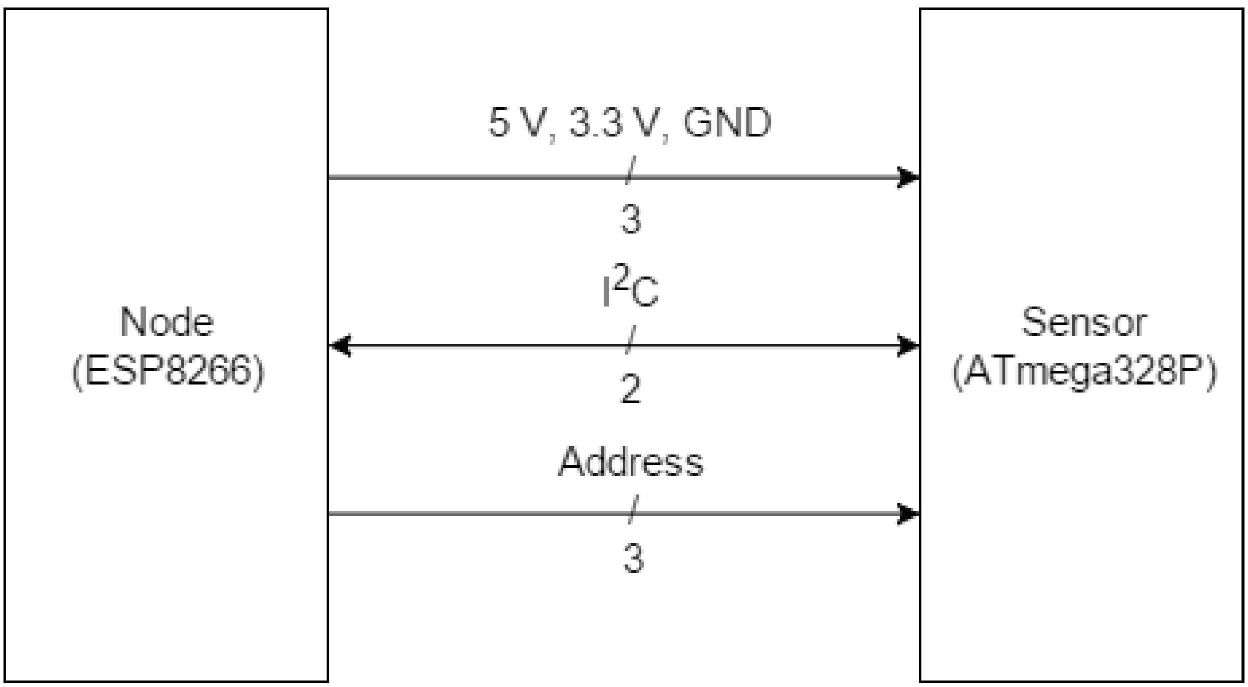

Figure 6.1: The connections between the node and sensor module

6.2 Touchscreen Protocol

The touchscreen communicates with the ATmega328P over a bi-directional parallel port according to the five status bits as diagrammed in Figure 6.2 and Table 6.2. To write data to the LCD, the ATmega328P brings CS low to enable the touchscreen. It then asserts CD low for sending a command or high for sending data followed by placing the data to transmit on the data bus. Finally, the ATmega329P brings WR low for a few clock cycles and back high. To read data from the LCD, the ATmega328P brings CS low to enable the touchscreen, brings RD low for a few clock cycles, and brings RD back high. It then reads the byte off the data bus.

Table 6.2: Touchscreen Physical Interface

Pin Symbol	Pin Name	Pin Explation
$\overline{\text{CS}}$	$\overline{\text{Chip Select}}$	Enables touchscreen communications when low
CD	$\overline{\text{Command}}$/Data	0: Command, 1: Data
WR	Write	Writes data to the touchscreen on falling edge
RD	Read	Reads data from the touchscreen on falling edge
$\overline{\text{RST}}$	$\overline{\text{Reset}}$	Resets the touchscreen when low
D7:D0	LCD Data	A byte transmitted between ATmega328P and touchscreen

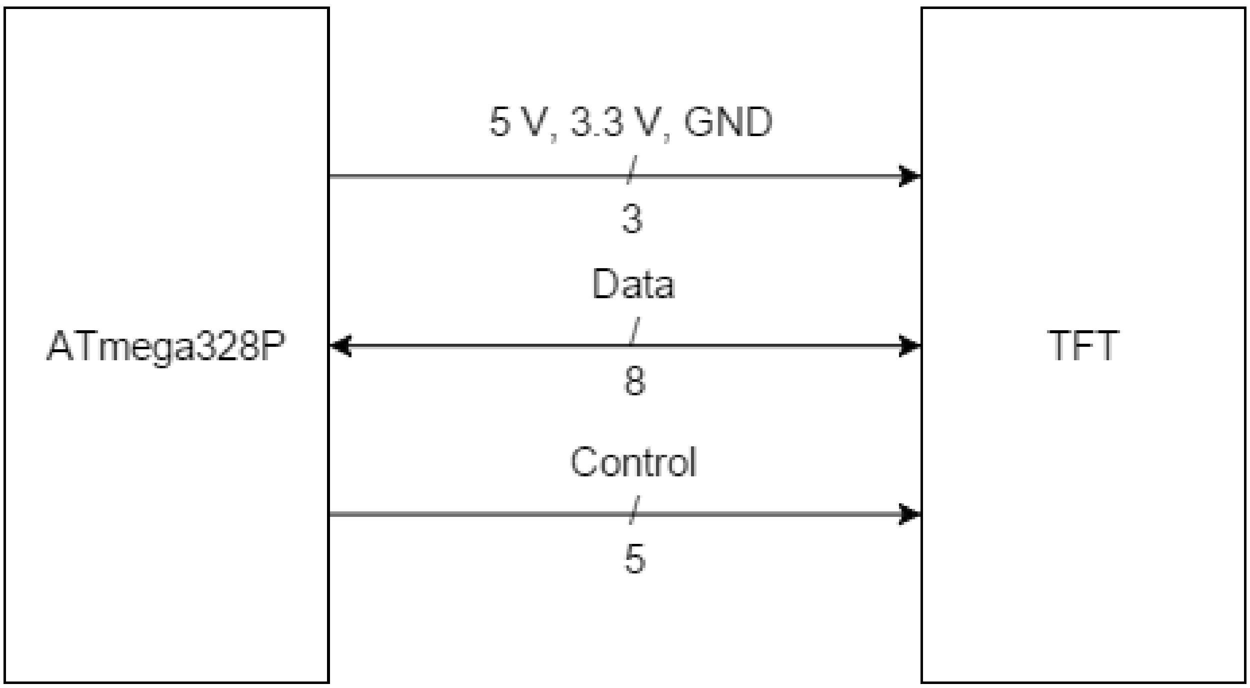

Figure 6.2: The connections between the touchscreen and ATmega328P

6.3 DHT22 Protocol

The DHT22 communicates over a digital, bi-directional, single pin bus that is normally high through the connections made in Figure 6.3. To begin a transmition, the ATmega first brings the bus low for at least 1 ms by configuring the pin as a digital output with a value of zero. It then configures the pin as a digital input with a weak pullup. 40 µs later, the DHT22 brings the bus low for 80 µs and high for 80 µs to indicate that it is about to begin transmission as shown in Figure 6.4. The DHT22 then brings the bus low for 50 µs and high for a variable amount of time. If the bus is high for 26 - 28 µs, the bit is interpretted as a "0" as shown in Figure 6.5. If the bus is high for 70 µs, the bit is interpretted as a "1" as shown in Figure 6.6. After all 40 bits have been transmitted, the DHT22 lets the bus float high. Note that the highest bit is transmitted first.

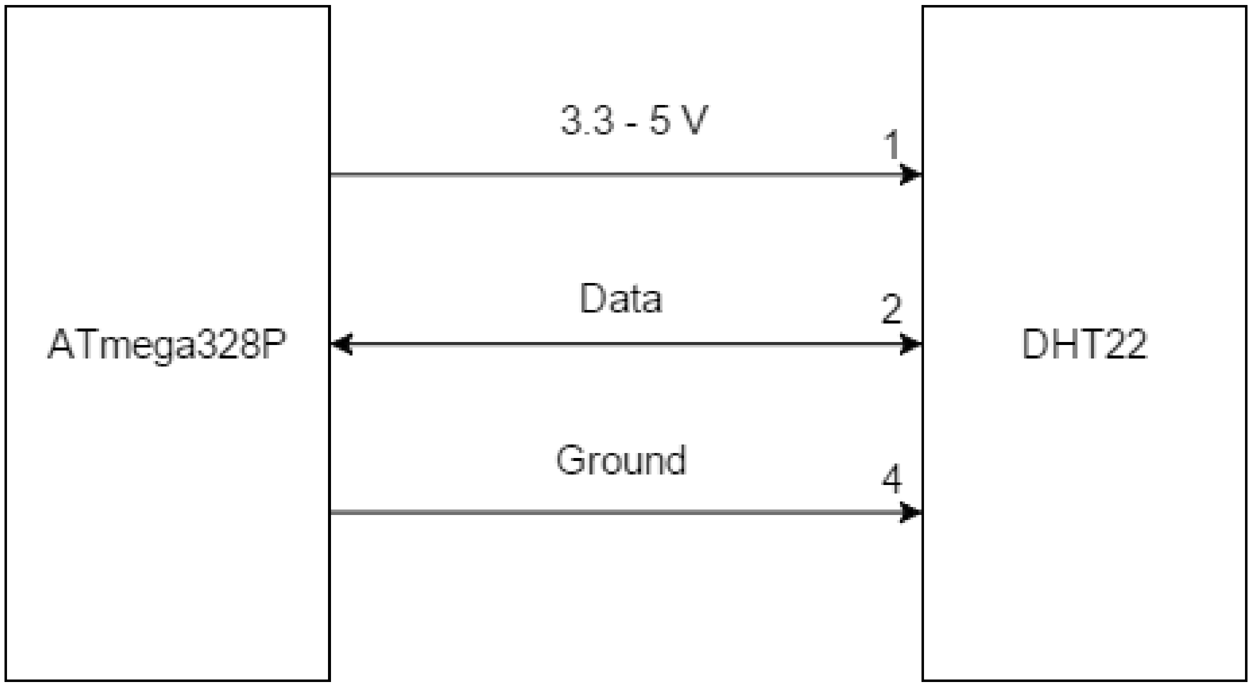

Figure 6.3: The connections between the sensor module and DHT22

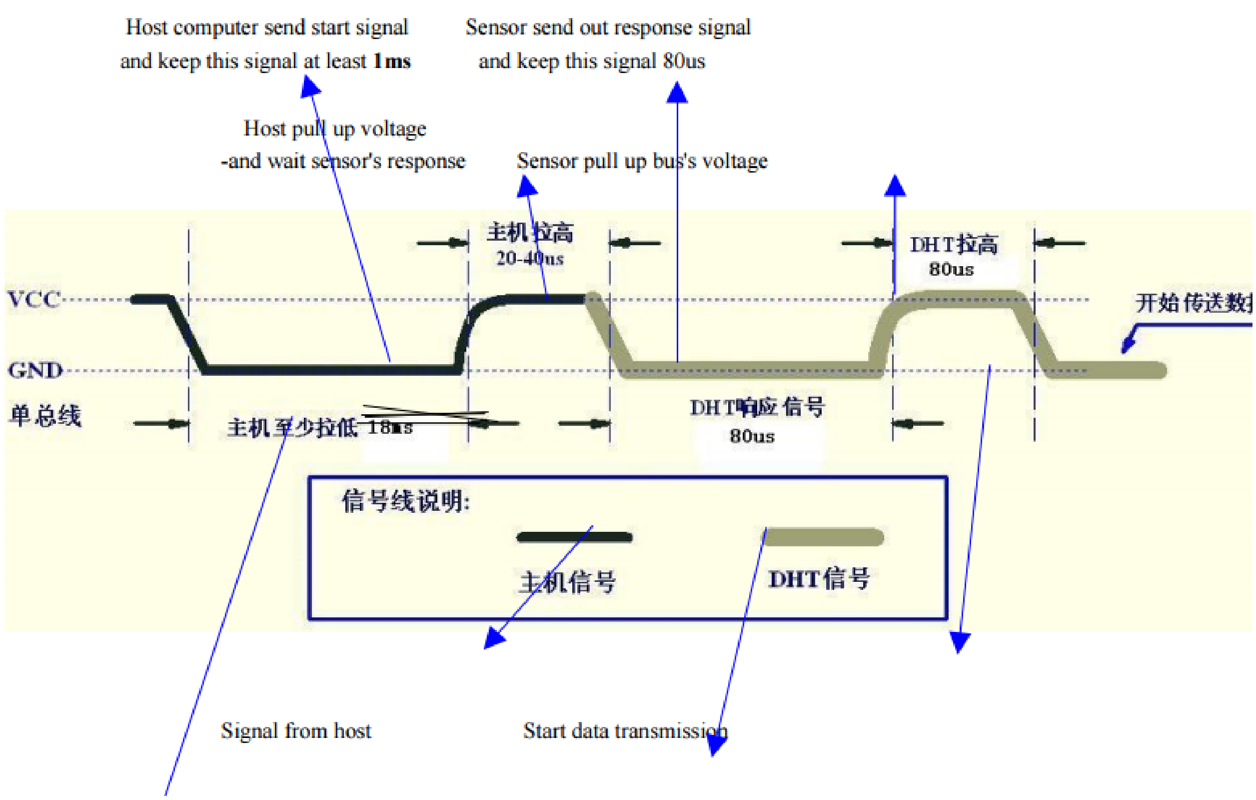

Figure 6.4: The initialization of a reading between the ATmega328P and DHT22

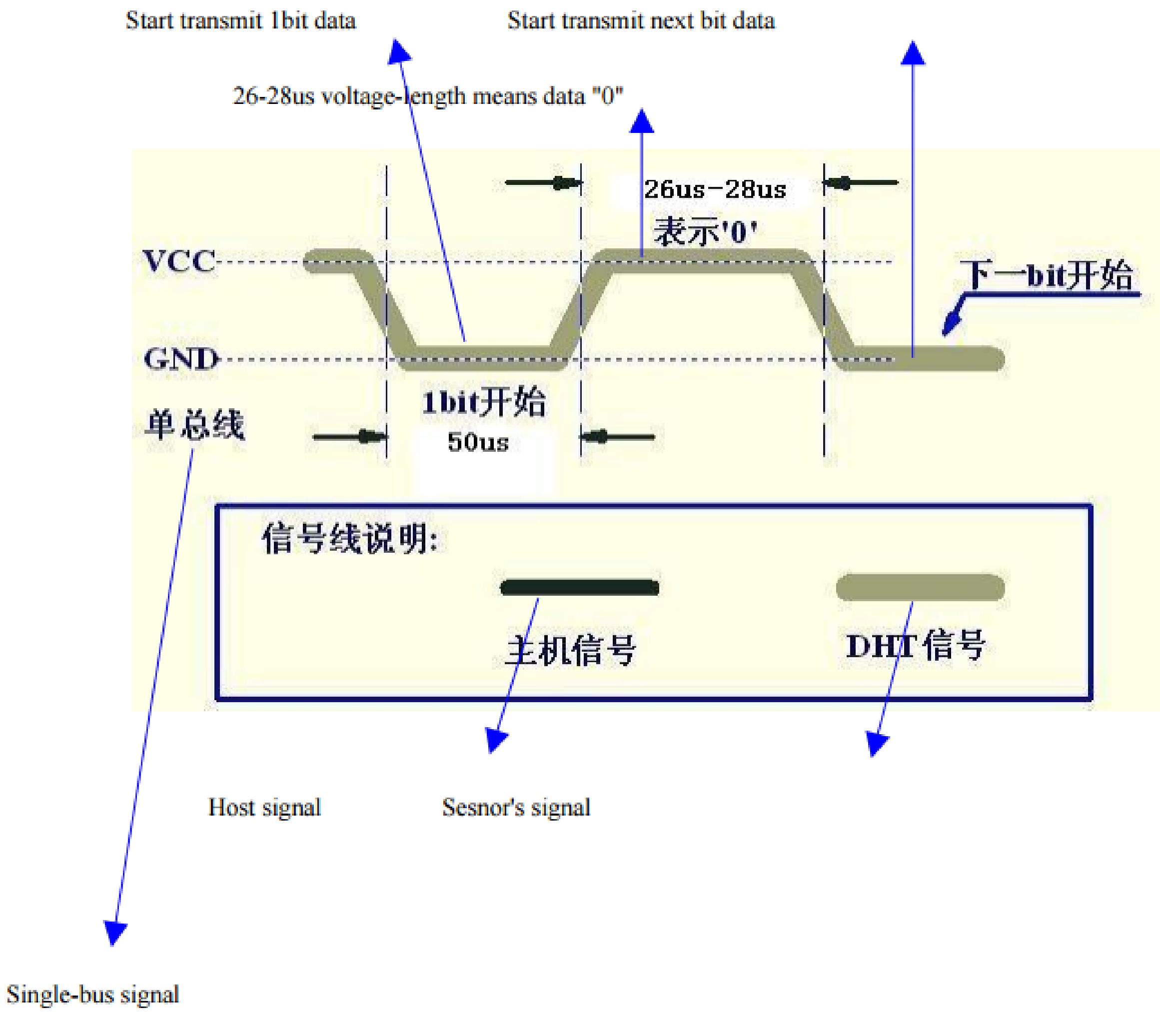

Figure 6.5: The DHT22 trasnmitting a "0"

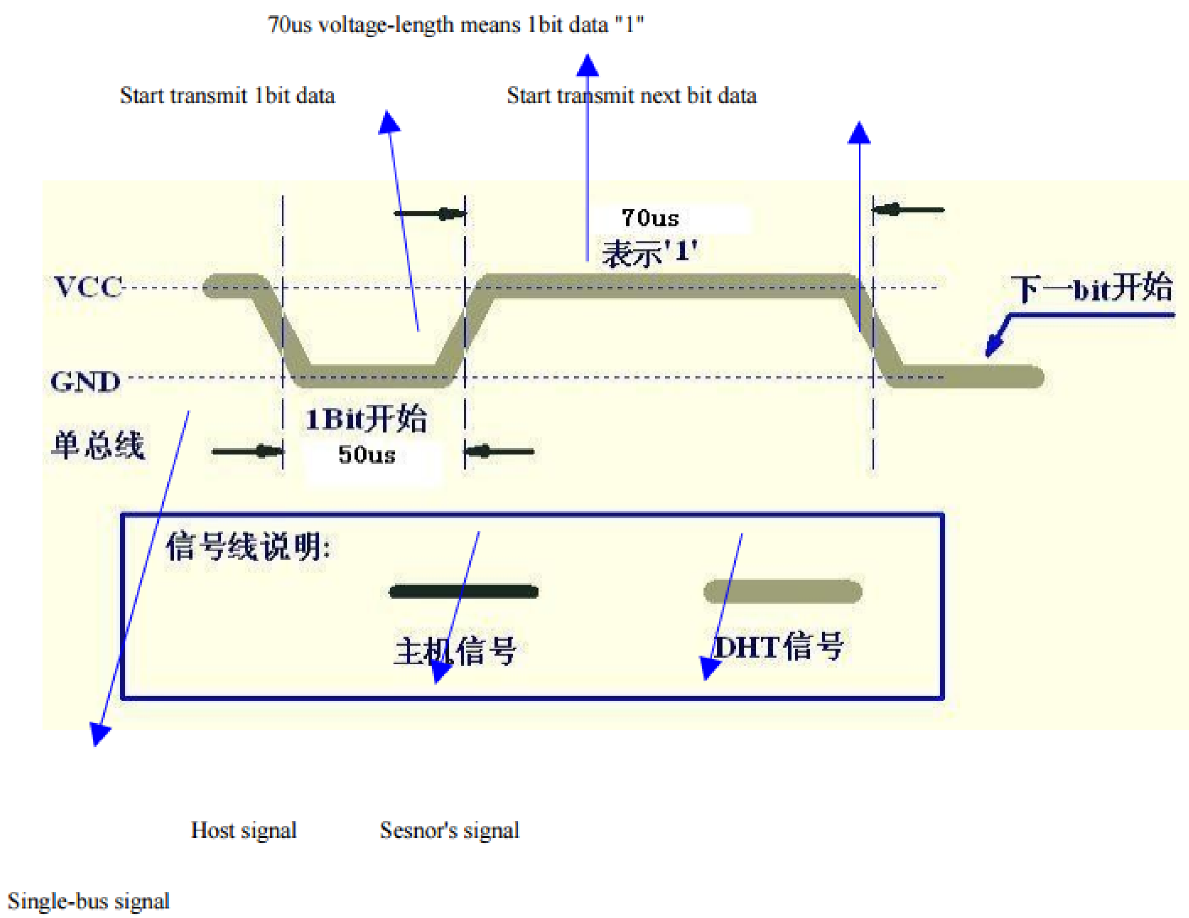

Figure 6.6: The DHT22 transmitting a “1”

6.4 AC Current Sensor

The current sensor uses a transformer to induce a current between 0 and 50 mA (RMS) at 60 Hz on the secondary coil. This current is passed through the RC network in Figure 6.7 to generate a sinisoid centered at 2.5 V oscillating between 0 V and 5 V. This wave is sampled on the ATmega328P's analog input channel to calculate the RMS current though the primary coil. With the 10 bit ADC on the ATmega328P, this translates to a 98 mA (RMS) resolution on the primary coil's current.

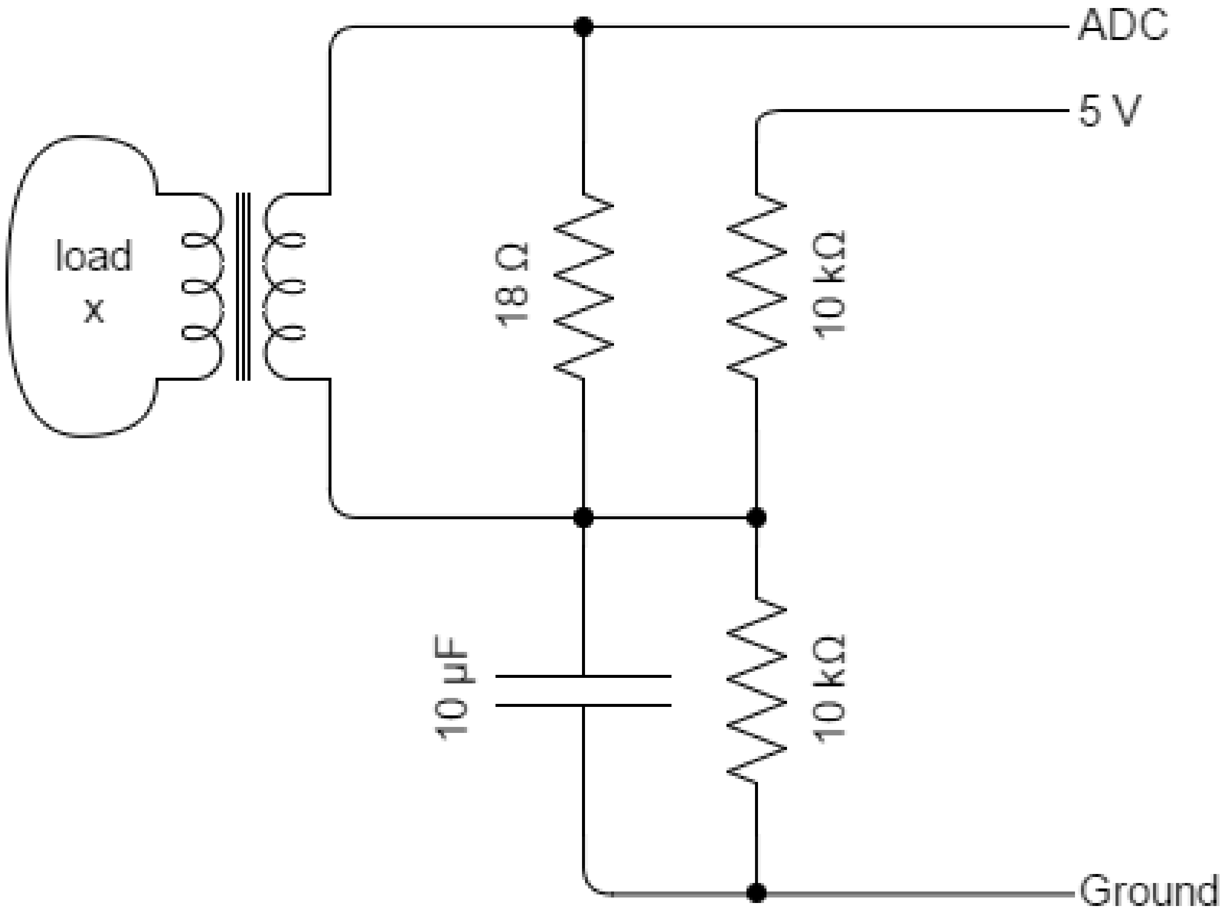

Figure 6.7: The connections between the sensor module and current sensor

6.5 Barometric Pressure Sensor

The barometric pressure sensor communicates with the ATmega328P through an I^2C bus as depicted in Figure 6.8. It does this by triggering the BMP180, reading the calibration data, reading the raw pressure reading, and calculating the true pressure data according to Figure 6.9. The calibration and status register addresses are recored in Figures 6.10 and 6.12, respectively. Once the barometric pressure has been calculated, the ATmega328P calculates the altitude from the relationship in Figure 6.11.

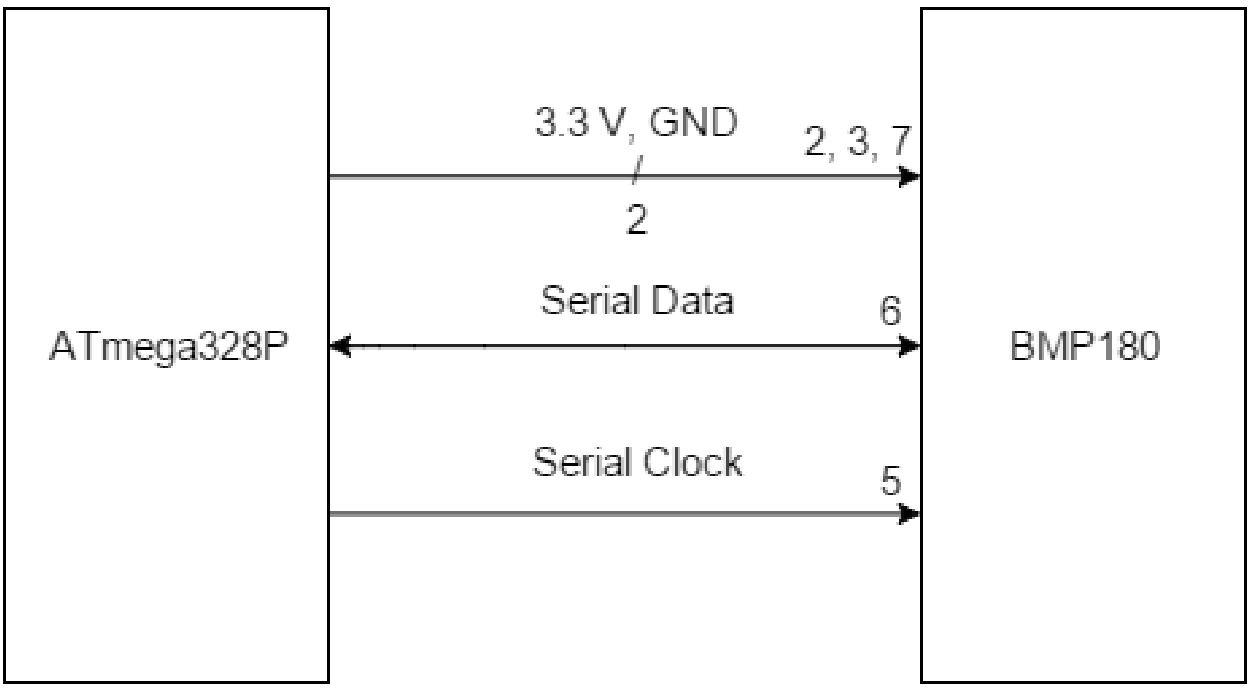

Figure 6.8: The interface between the ATmega328P and BMP180

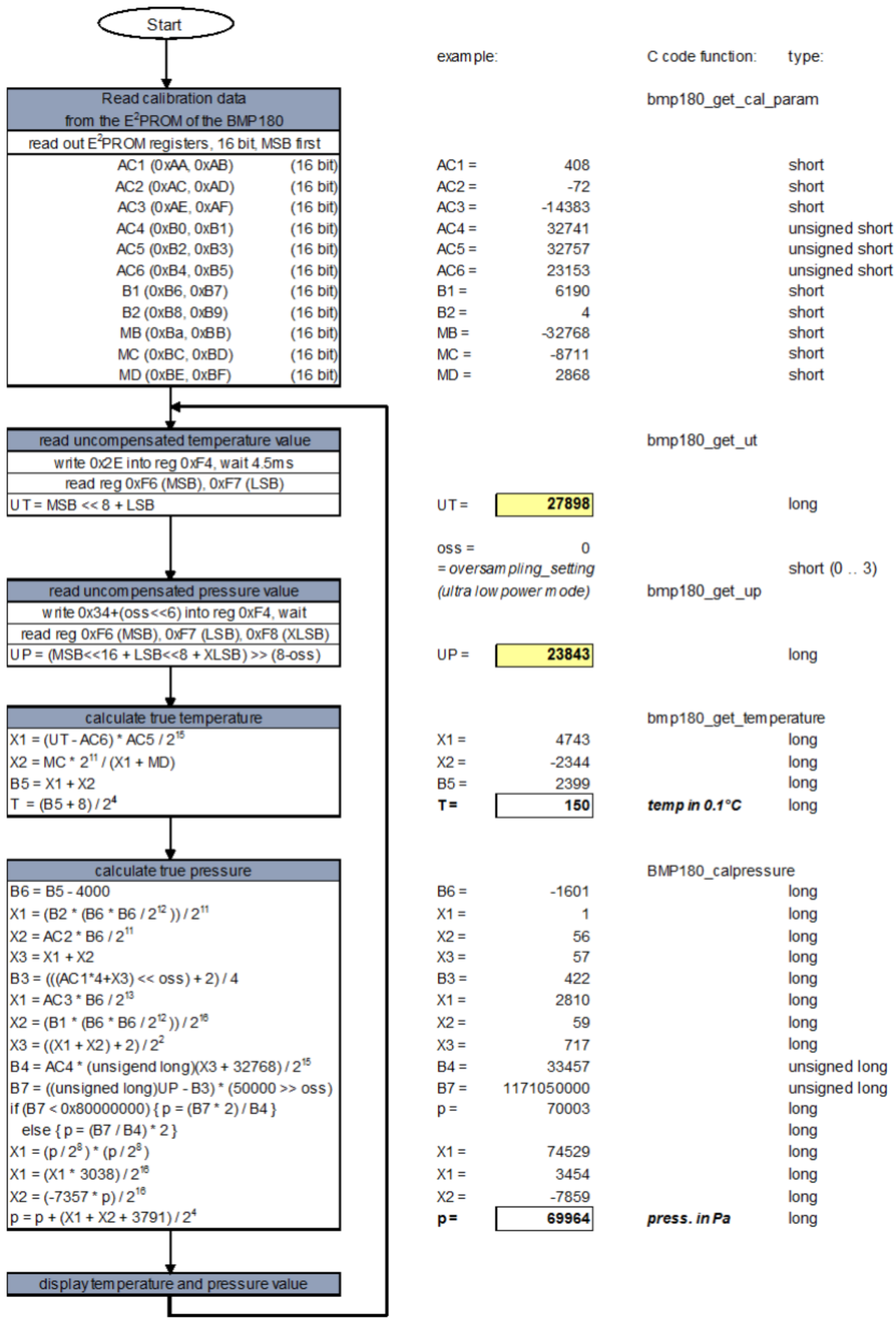

Figure 4: Algorithm for pressure and temperature measurement

Figure 6.9: The program flow for reading the BMP180

	BMP180 reg adr	
Parameter	MSB	LSB
AC1	0xAA	0xAB
AC2	0xAC	0xAD
AC3	0xAE	0xAF
AC4	0xB0	0xB1
AC5	0xB2	0xB3
AC6	0xB4	0xB5
B1	0xB6	0xB7
B2	0xB8	0xB9
MB	0xBA	0xBB
MC	0xBC	0xBD
MD	0xBE	0xBF

Figure 6.10: The calibration registers for the BMP180

$$\text{altitude} = 44330 * \left(1 - \left(\frac{p}{p_0}\right)^{\frac{1}{5.255}}\right)$$

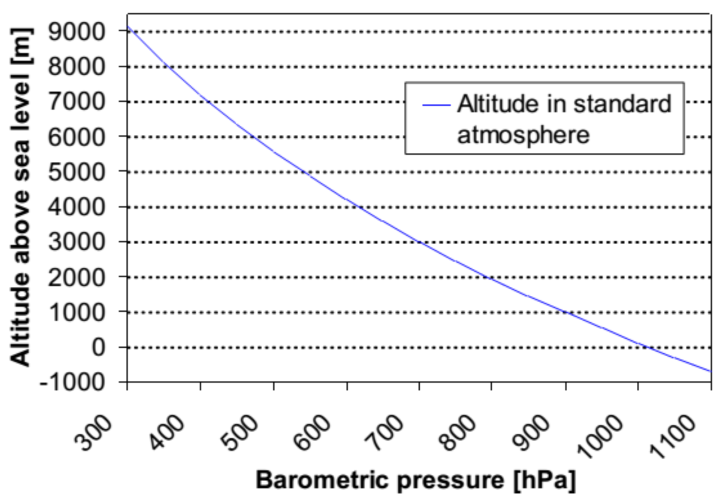

Figure 6.11: The relationship between barometric pressure and altitude

Register Name	Register Adress	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Reset state
out_xlsb	F8h	adc_out_xlsb<7:3>					0	0	0	00h
out_lsb	F7h	adc_out_lsb<7:0>								00h
out_msb	F6h	adc_out_msb<7:0>								80h
ctrl_meas	F4h	oss<1:0>		sco	measurement control					00h
soft reset	E0h	reset								00h
id	D0h	id<7:0>								55h
calib21 downto calib0	BFh *down* to AAh	calib21<7:0> *down* to calib0<7:0>								n/a

Registers:	Control registers	Calibration registers	Data registers	Fixed
Type:	read / write	read only	read only	read only

Figure 6.12: The status registers of the BMP180

6.6 Brightness Sensor

The brightness sensors connect to the ATmega328P's ADC channels where a higher analog voltage represents a brighter room. There are six brightness sensors on each sensor module, allowing for light detection in six directions as shown in Figure 6.13.

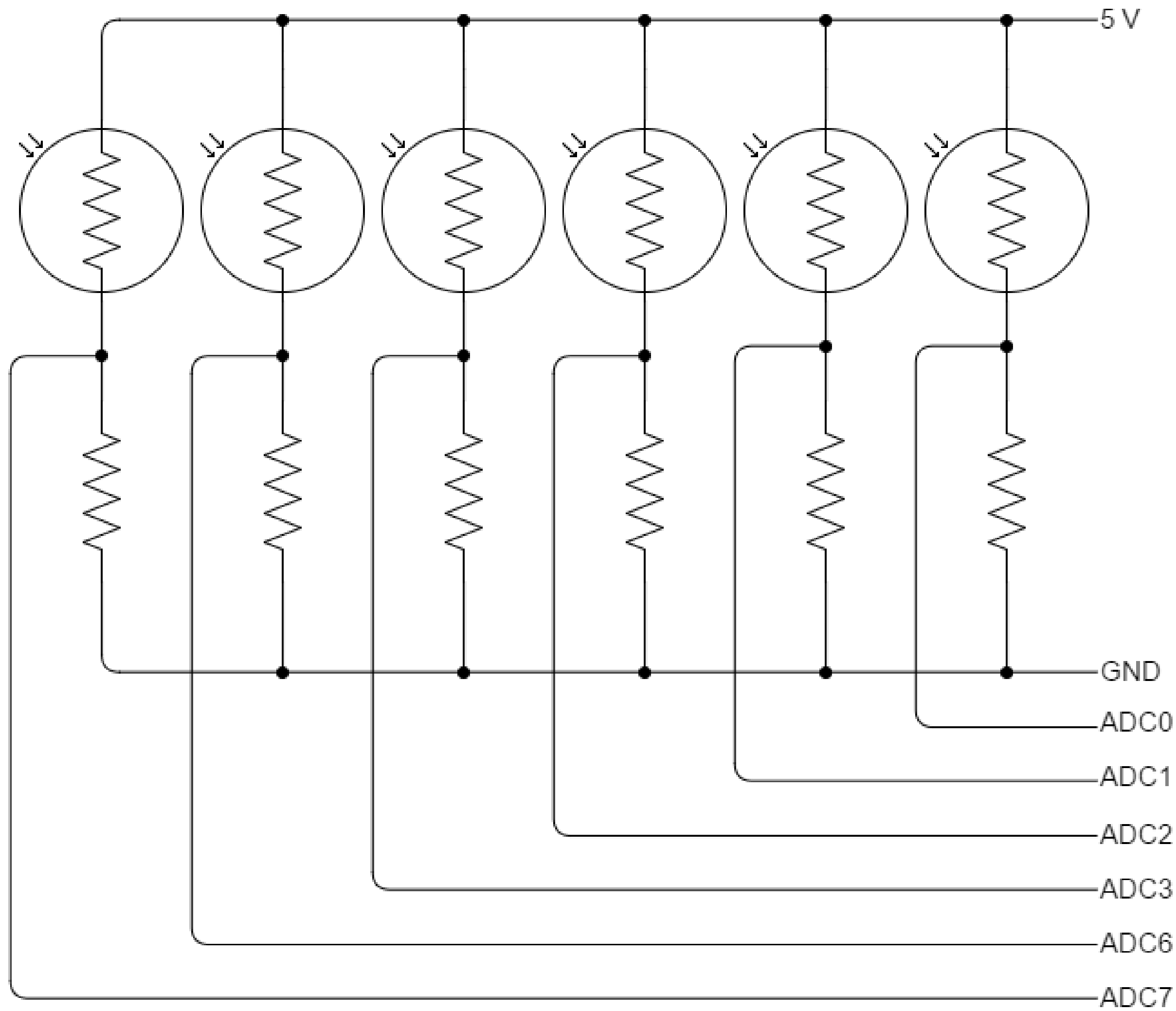

Figure 6.13: The interface between the brightness sensors and ATmega328P

6.7 Audio Sensor

The audio sensor

7 INPUT/OUTPUT DIAGRAM

The logical datapath for the project is diagramed in Figure 7.1. The network diagram for connecting different physical modules are presented in Figure 7.2. The inputs/outputs are summarized below.

- Sensor Data
  - Temperature
  - Humidity
  - Barometric Pressure / Elevation
  - Current / Energy usage
  - Brightness
  - Sound
- Control Devices
  - Air Conditioner / Heater
  - LEDs
  - Outlets
  - Lights
- Node Configuration
  - Sampling Rate
  - Precision
  - Control Conditions
  - Network
  - Link to Account
- Plotted Sensor Data
  - Organize sensor data graphically for user
- Sensor Data Analytics
  - Averages
  - Cummulative Totals
- Network Status
  - Nodes online
  - Last sensor readings

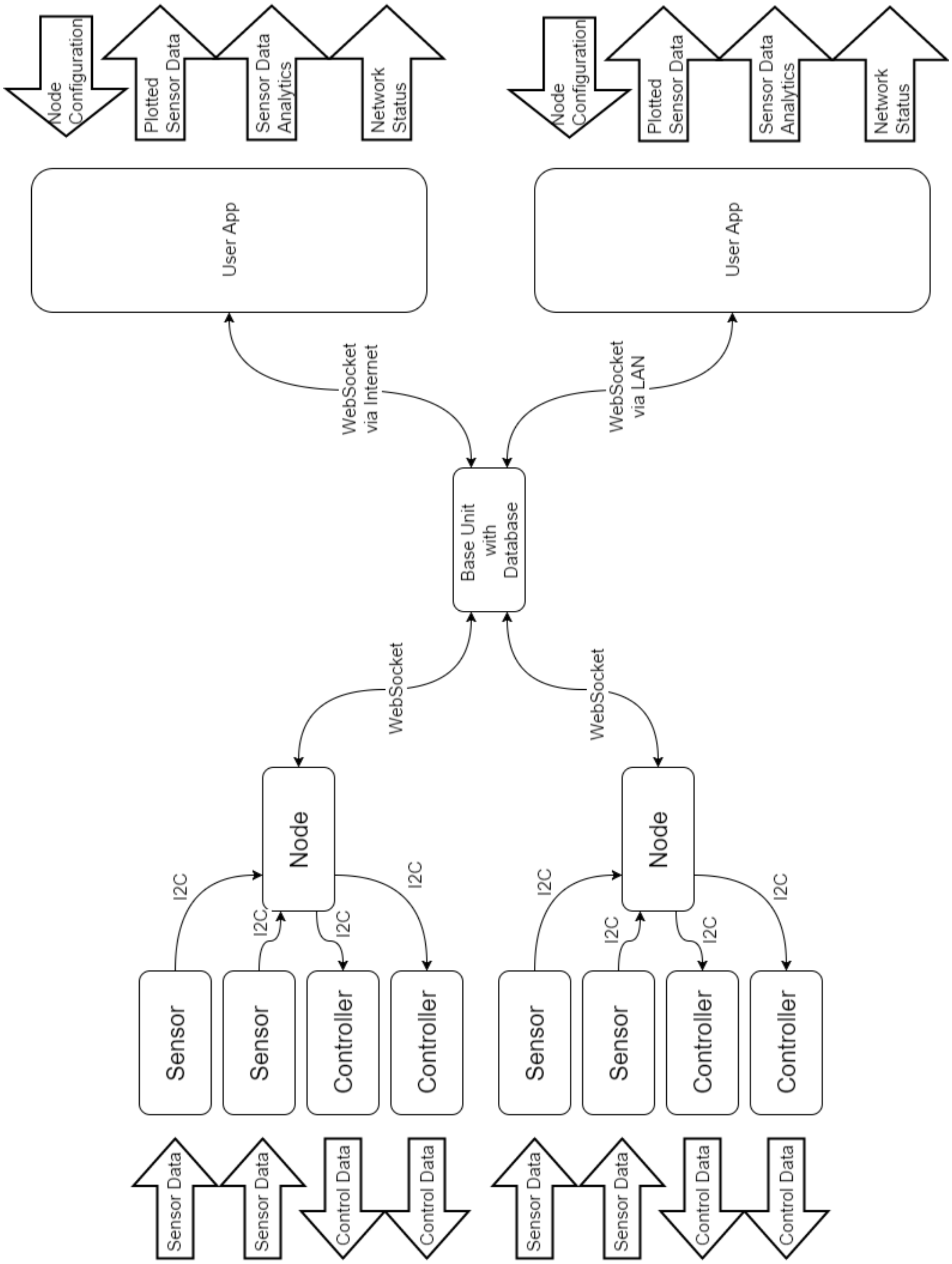

Figure 7.1: The Logical Datapath

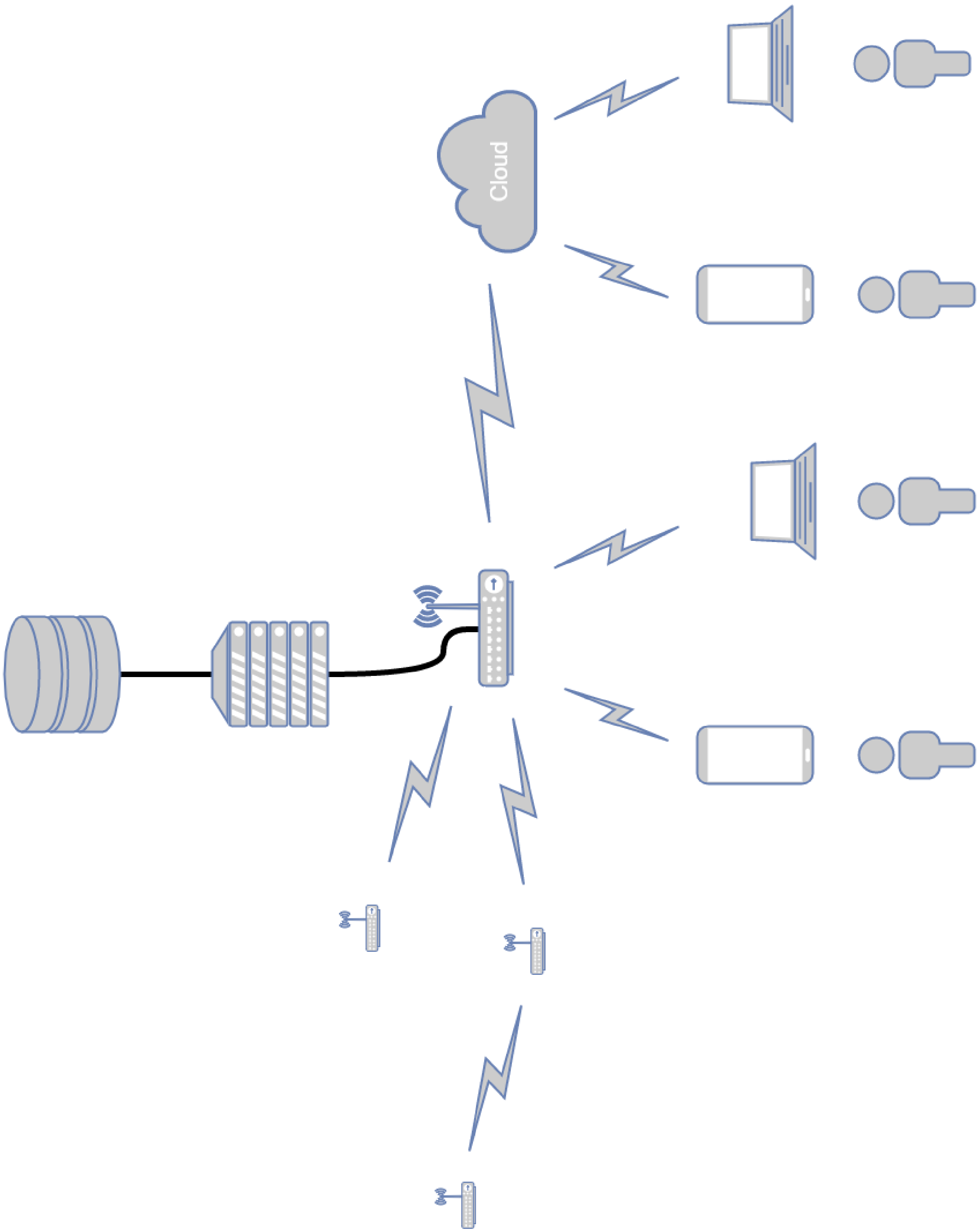

Figure 7.2: The Physcical Datapath

8 PROJECT TIMELINE

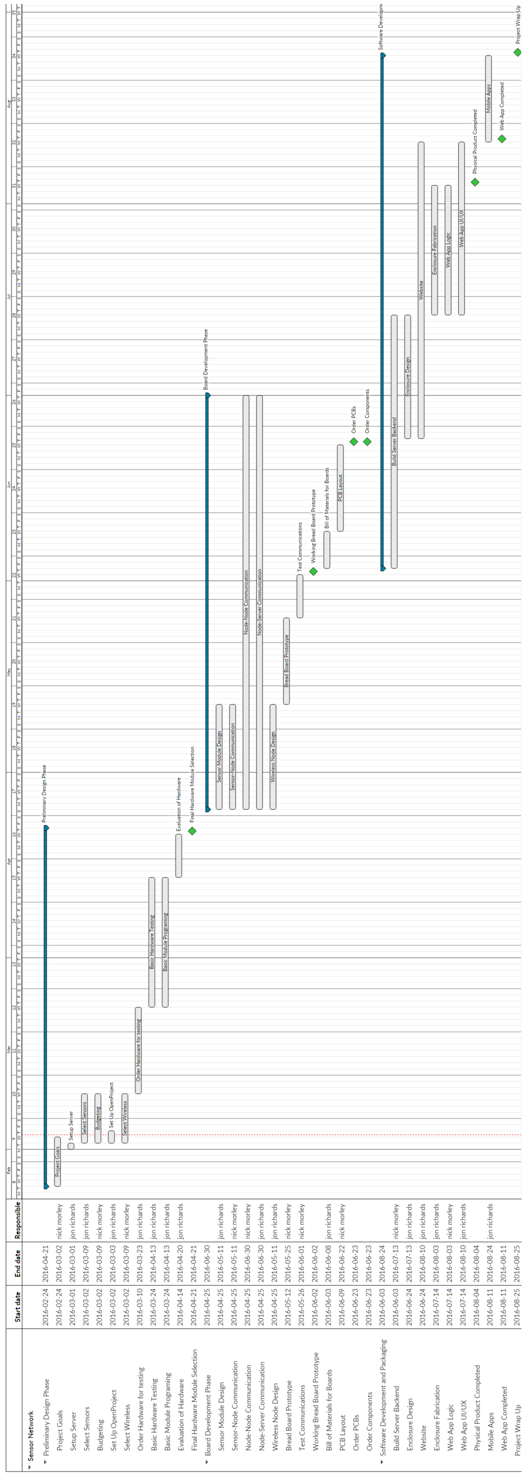

Figure 8.1: Gant Chart for Project Timeline