




# Supply and Demand Parking Meter

Team 33  
Adam Barbato and Nick Johanson



# Motivation

- It's hard to find parking in population dense areas
- Most solutions involve adding more parking, but it never fixes the issue
- We believe the problem is with the price, not the supply

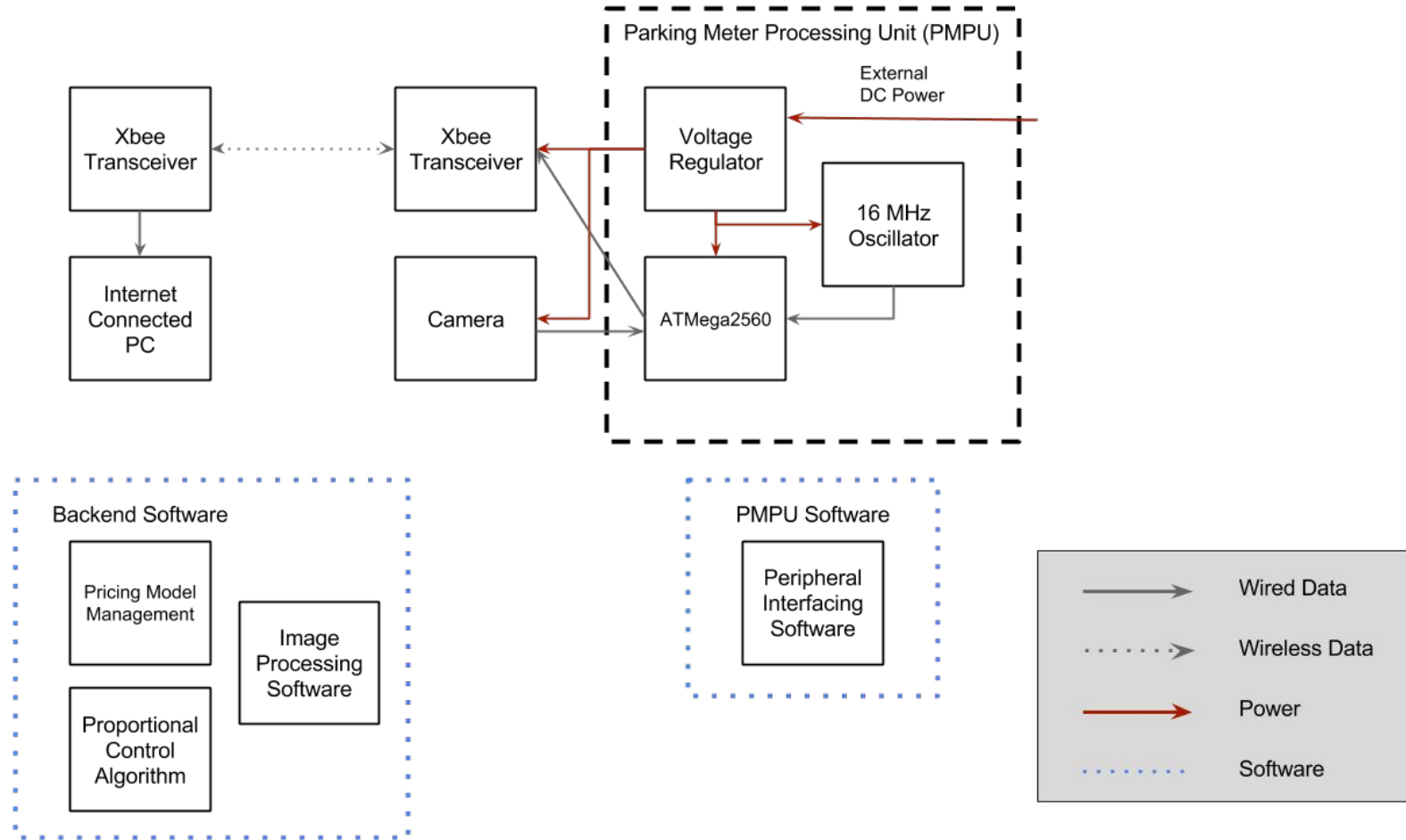


# Objective

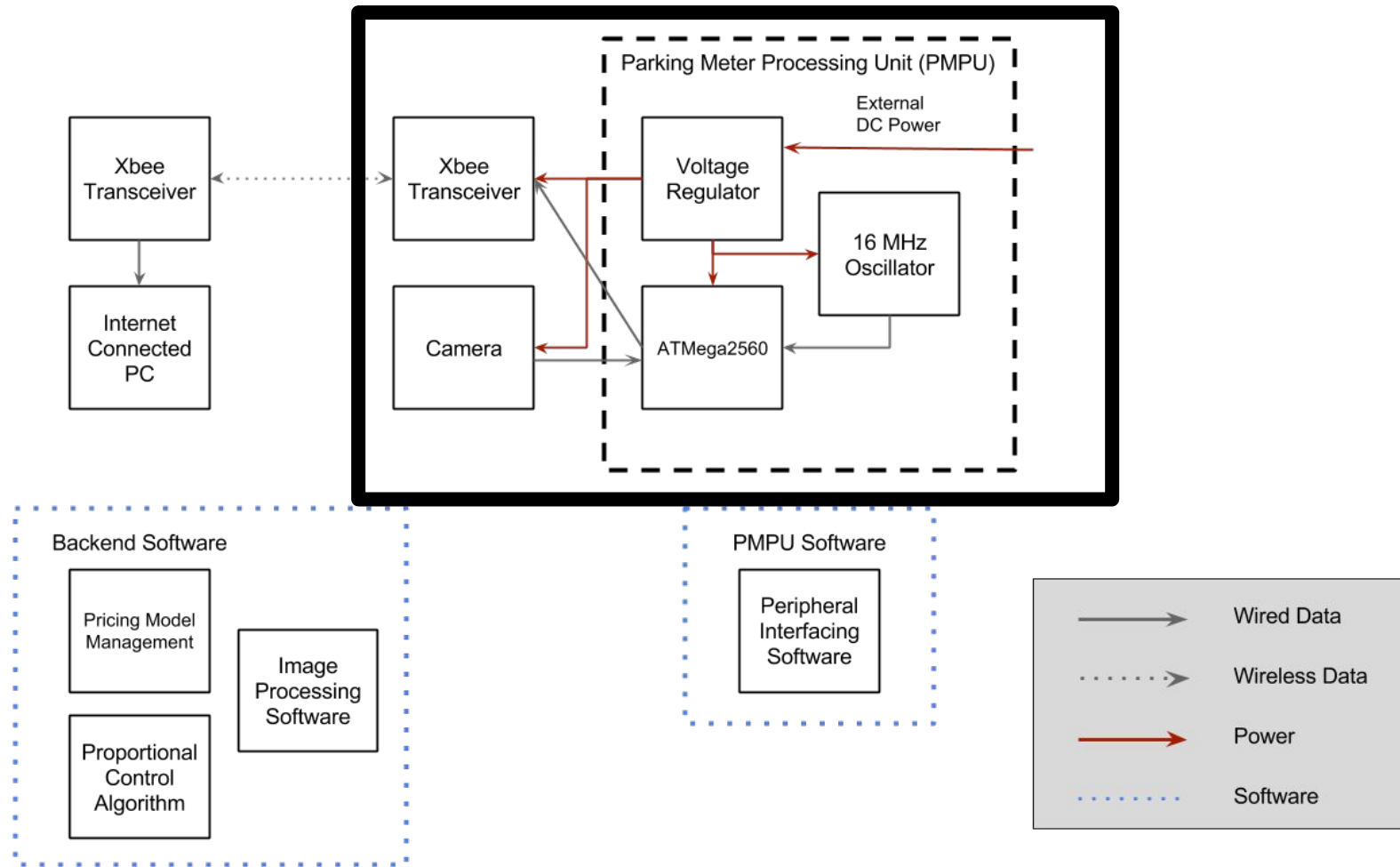
- Need a system that can dynamically adjust parking prices based on current and predicted demand
- Must be able to monitor car parking patterns
- See and learn from responses to a change in price

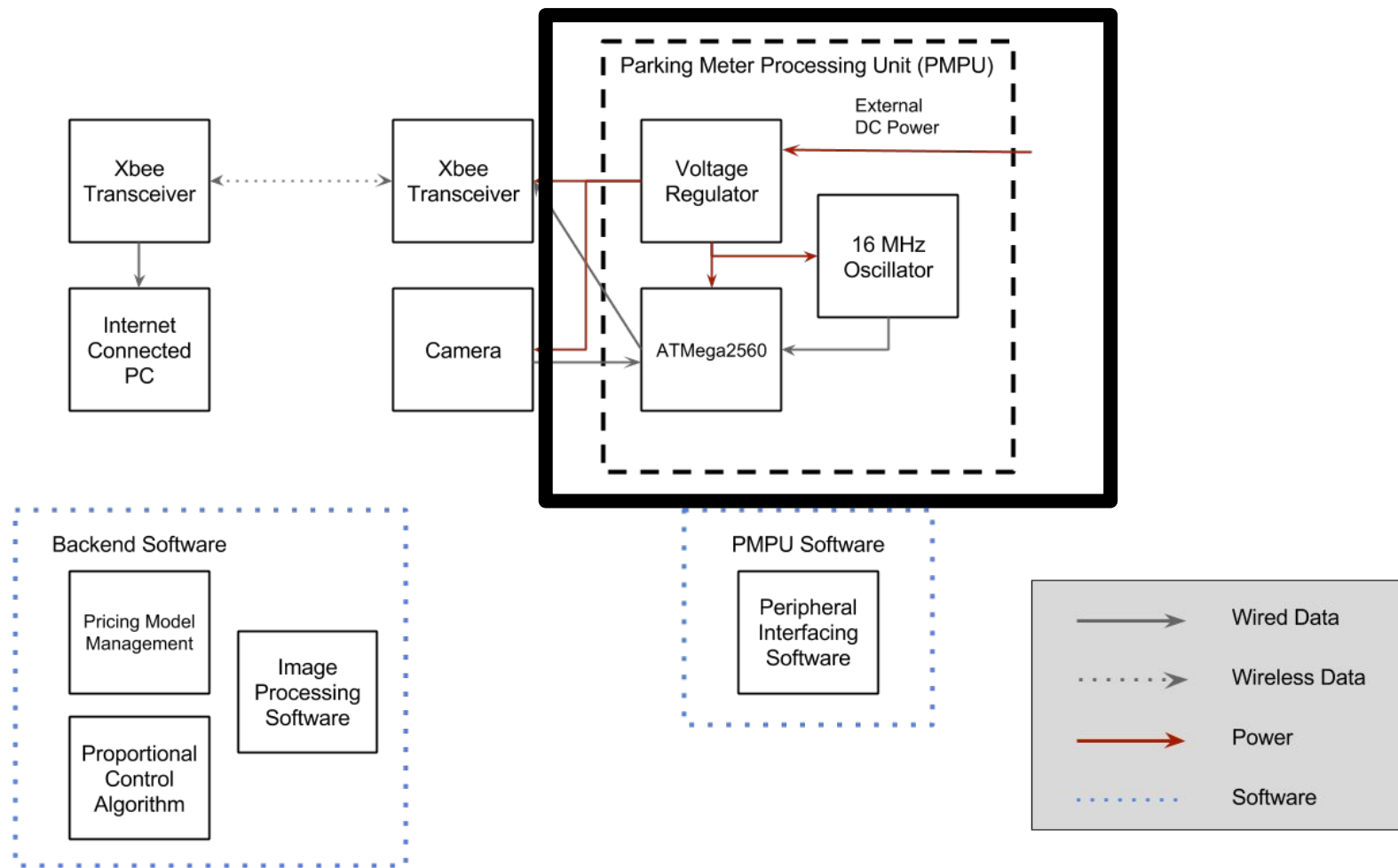
# Project Introduction

- Fully encapsulated system to record cars parked on street and figure out optimal price with two main components:
  - Computerized parking meter that can detect the number of cars present with Computer Vision and send data to backend
  - Backend software based on Control Theory to adjust prices based on demand and a simulated city block to test it
    - Attempts to locally optimize parking density in the hopes of achieving a global optimal result.



# Hardware

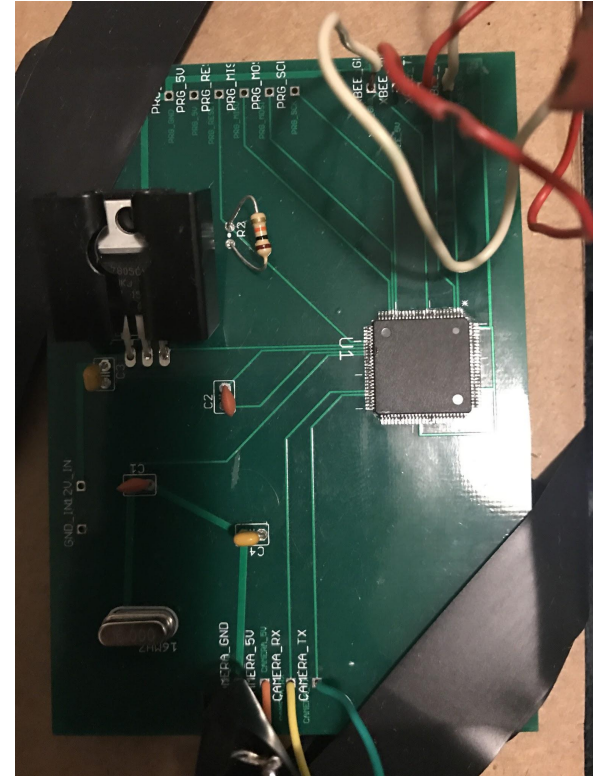


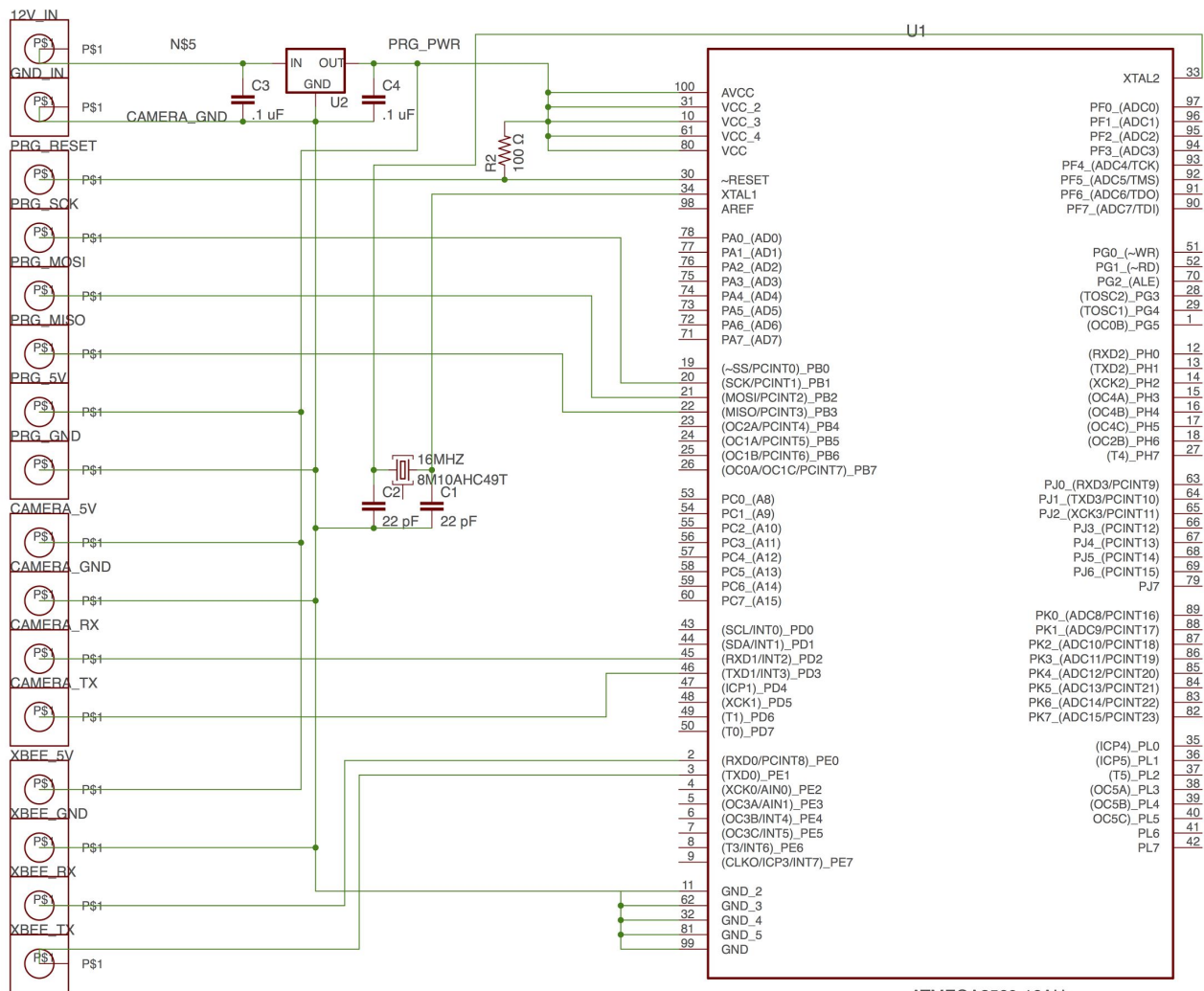




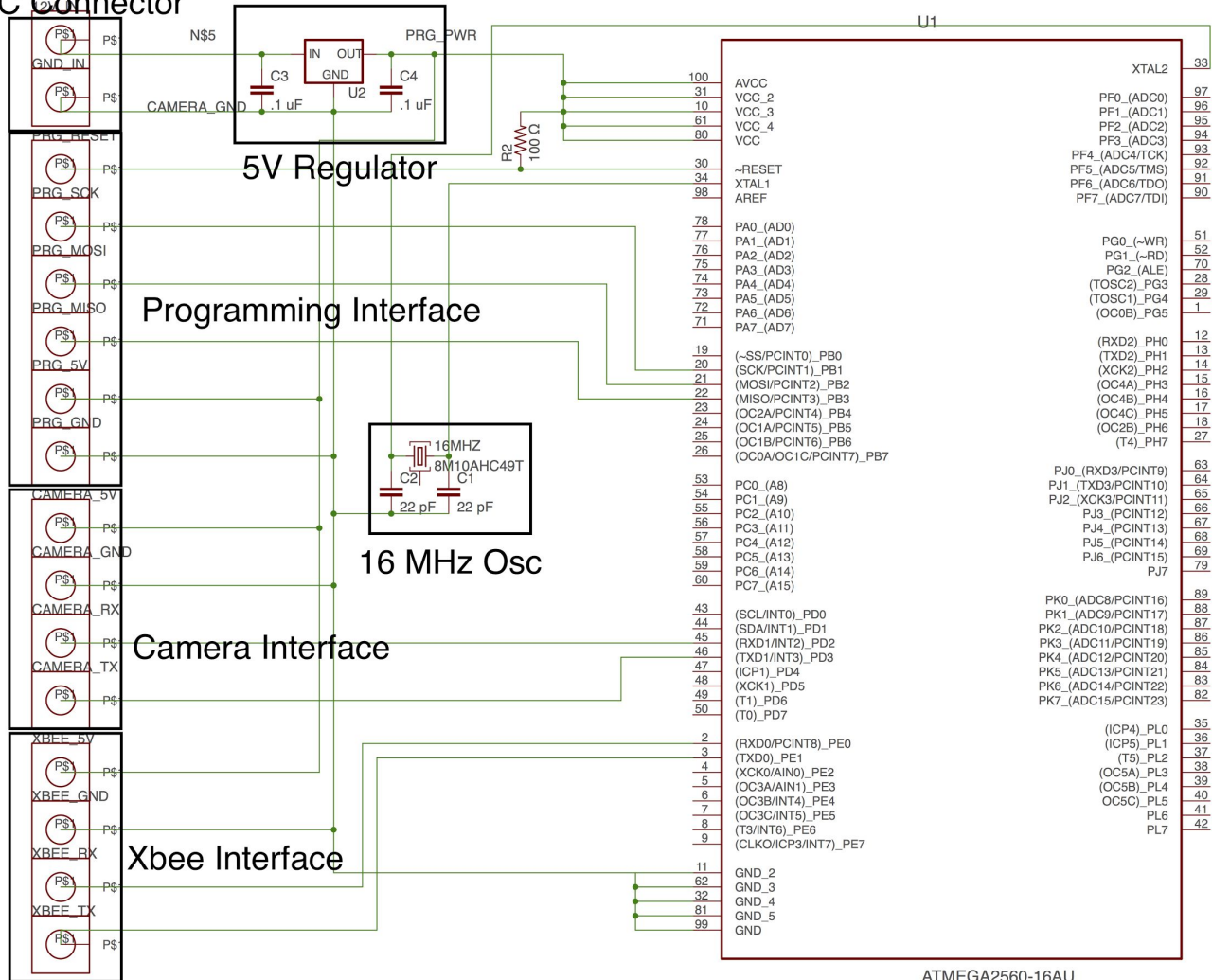
# Parking Meter Processing Unit (PMPU)

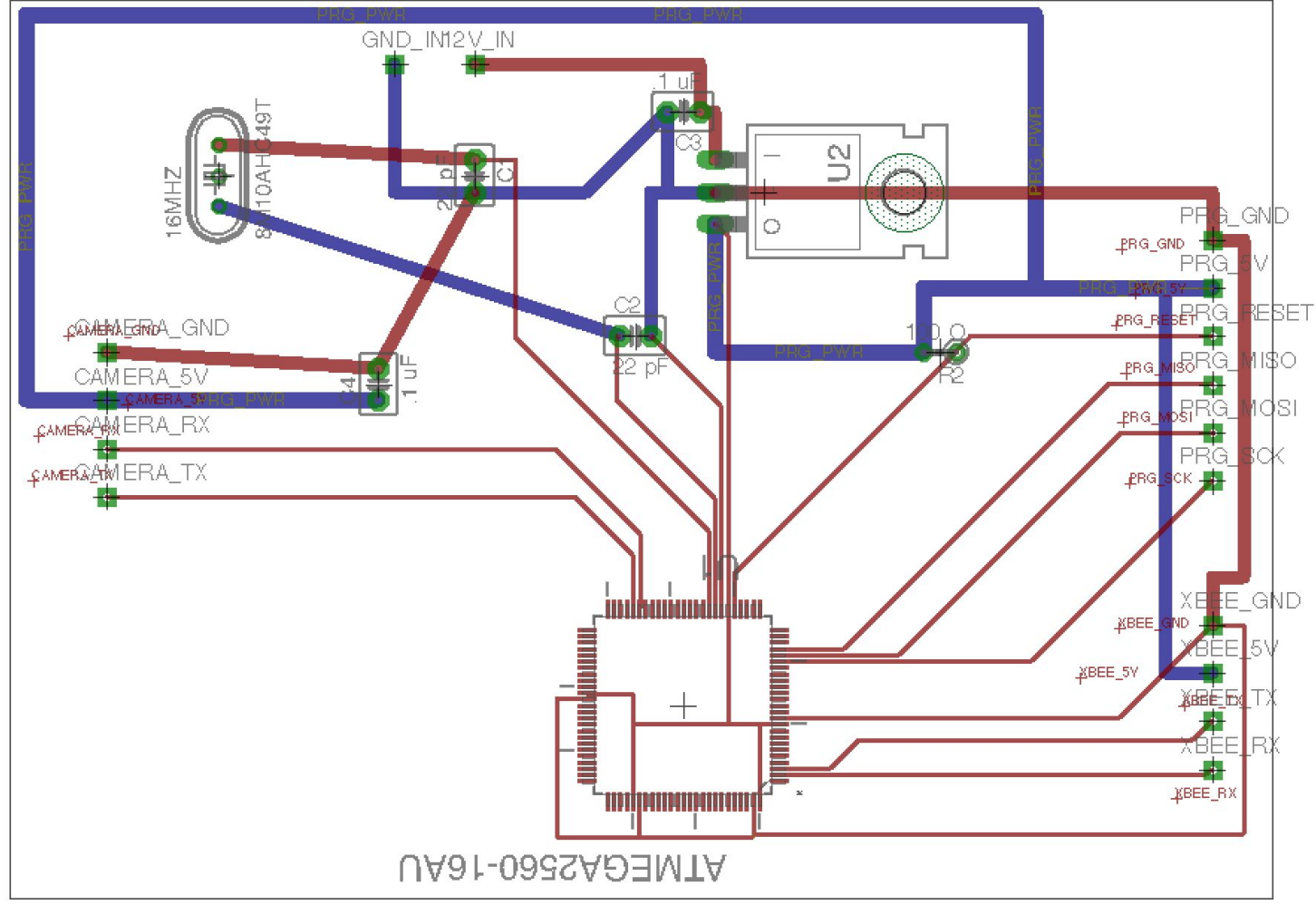
- Main hardware of project
- Three main components
  - 5V Voltage Regulator - take incoming 12V DC and regulate to 5V
  - Crystal Oscillator - produce a 16 MHz signal for ATmega
  - ATmega2560
    - Main processing power of PMPU
    - PCB must be designed and soldered correctly such that ATmega can be uploaded with and run arbitrary arduino code





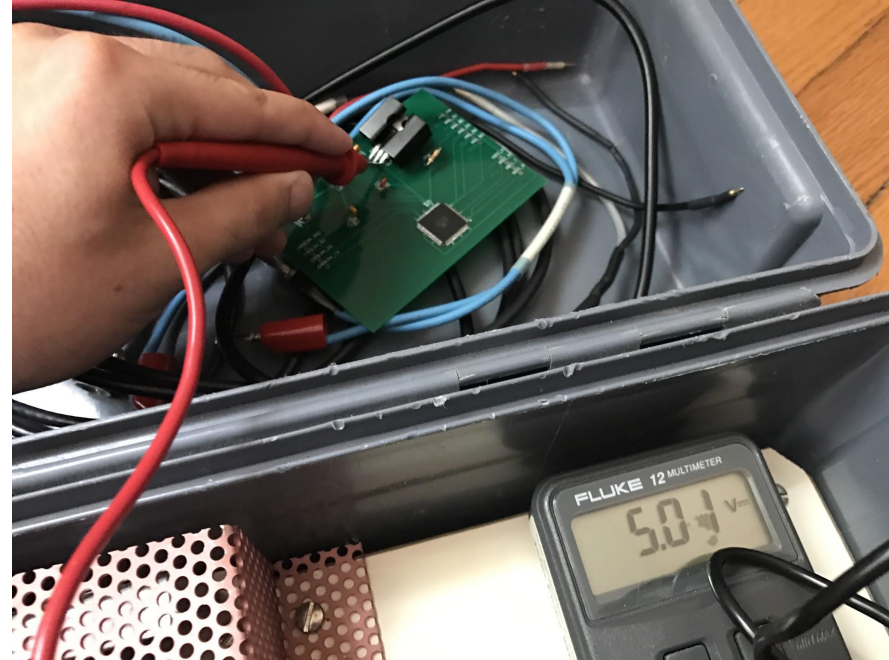
## DC Connector





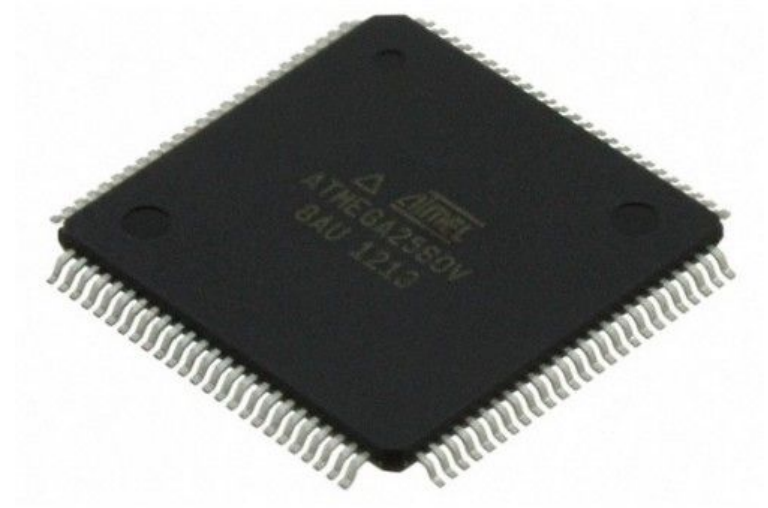
# PMPU Verifications

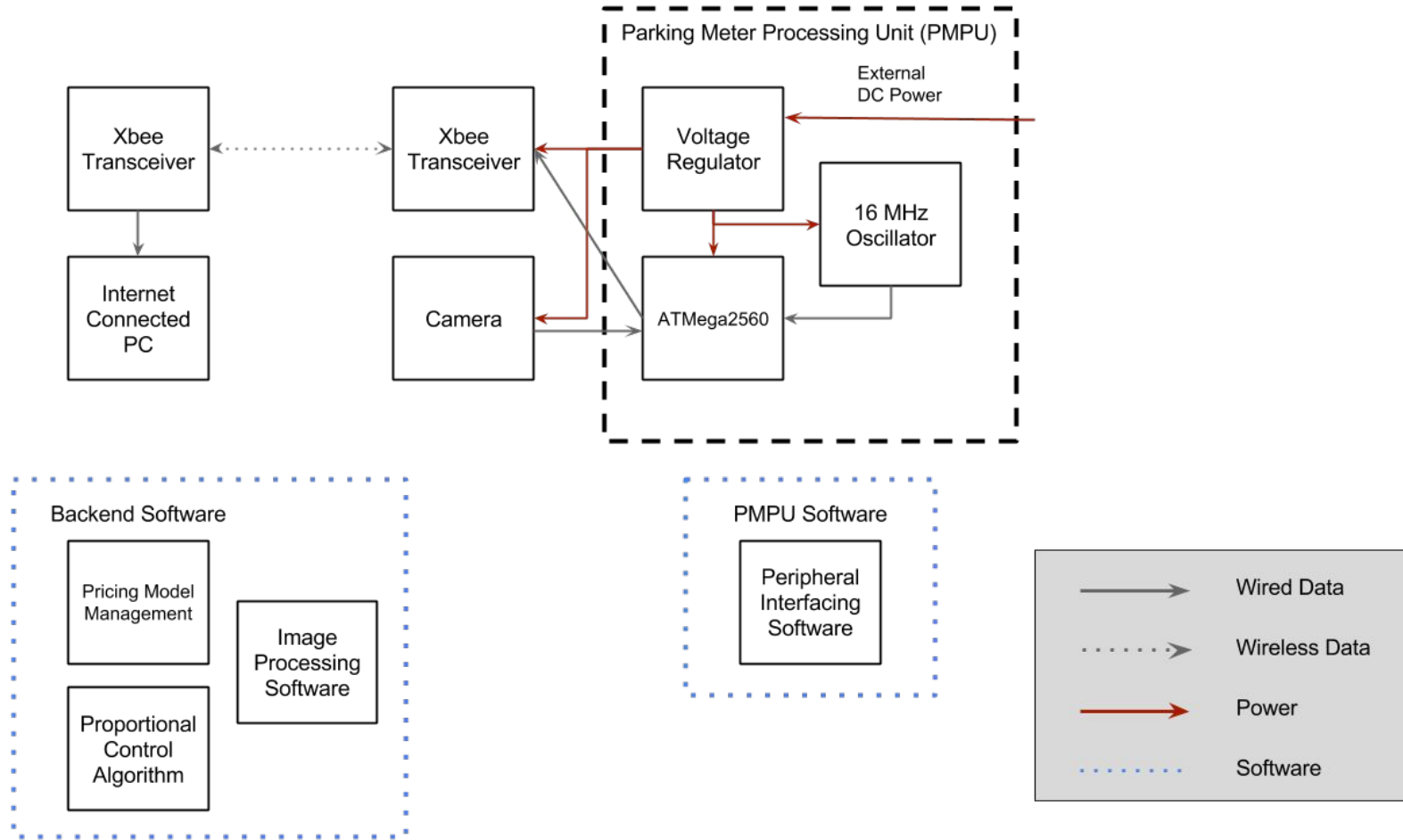
- 5V Voltage Regulator
  - Test shown on right
  - Regulated 12.07 V to 5.01 V
- 16 MHz Oscillator and ATmega2560
  - Proper functionality shown by uploading “Blink” sketch to the board
  - Voltage changed at same speed as Arduino Mega



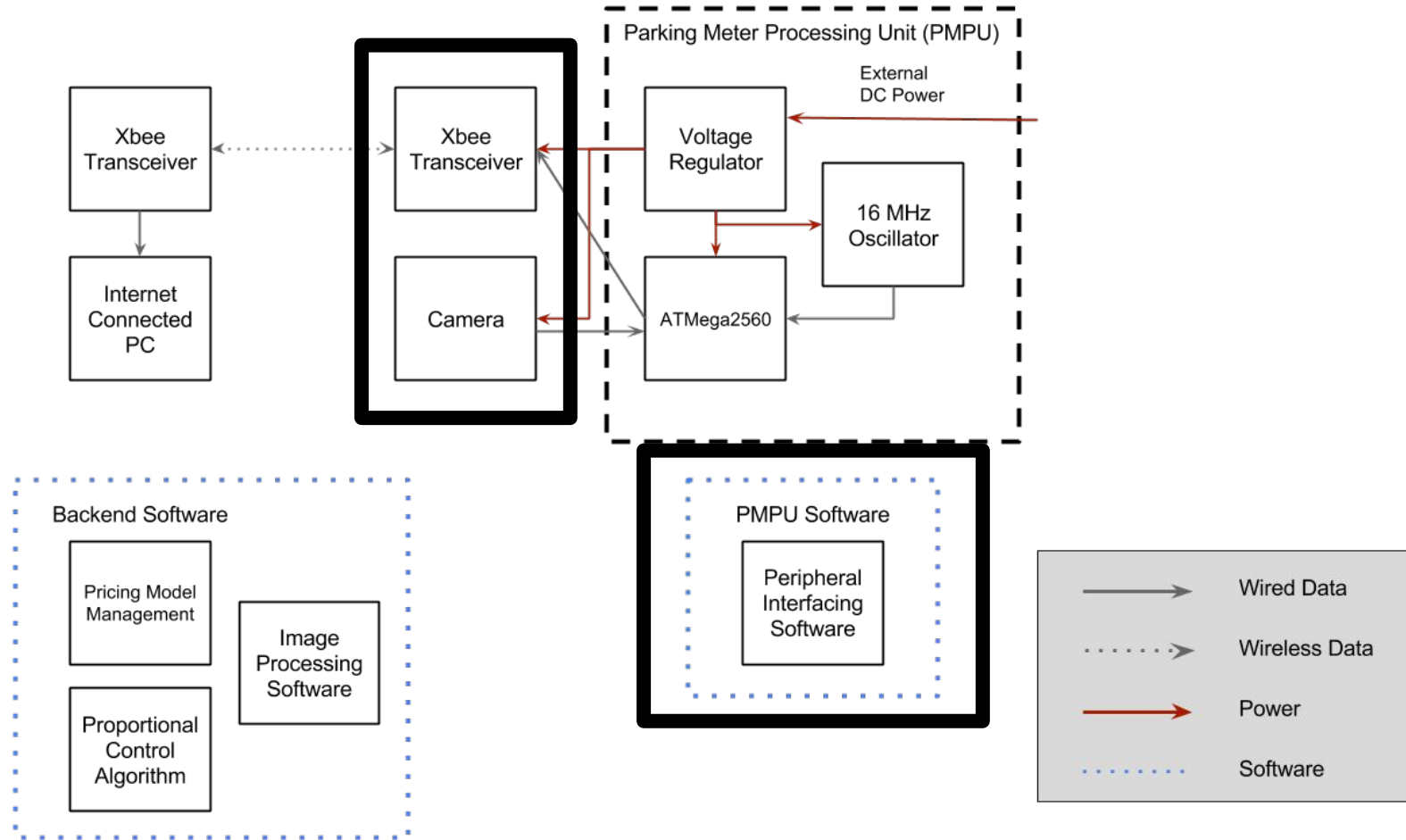
# Successes and Challenges

- No failed verifications
- PCB worked on first printing
- QFP (Quad Flat Package) was much harder to solder than expected
  - Much smaller than expected
  - Needed to use heavier techniques than provided in 445 lab











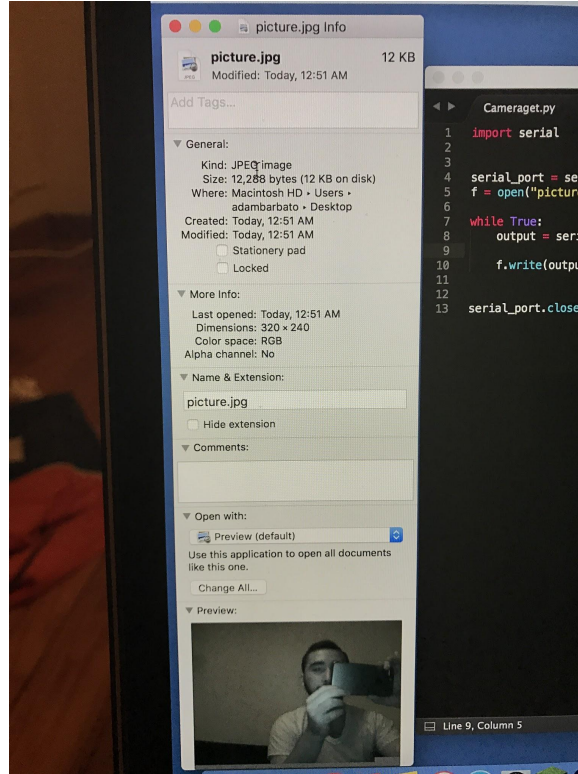
# Camera

- Weatherproof Serial TTL Arduino Camera, chosen for ease of use
- Main requirements on software that controls it



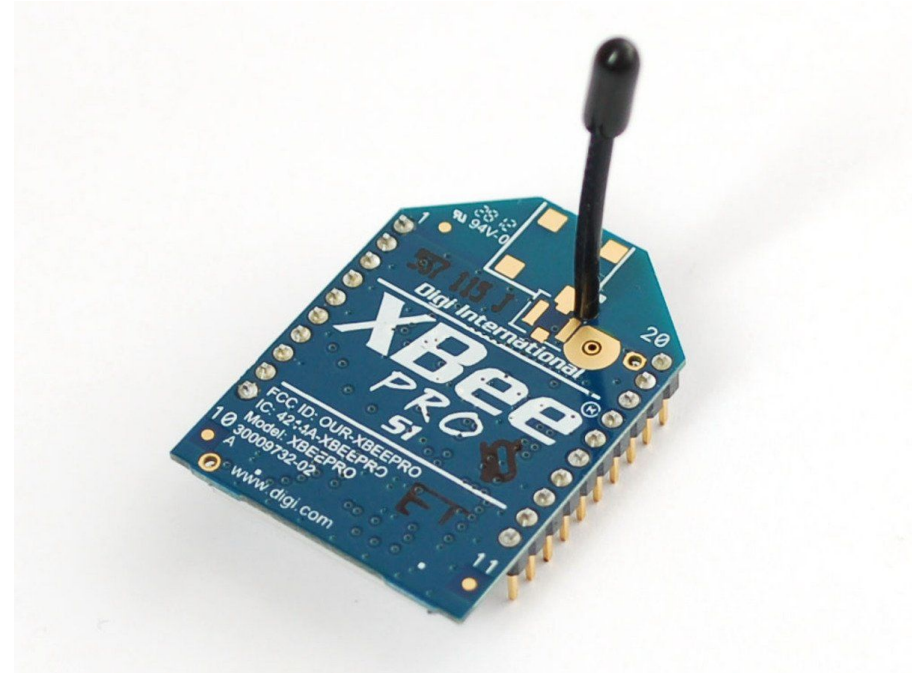
# Camera Requirements and Verification

- Must be able to capture two cars in Field of Vision
- Must be able to transmit QVGA picture when instructed by software



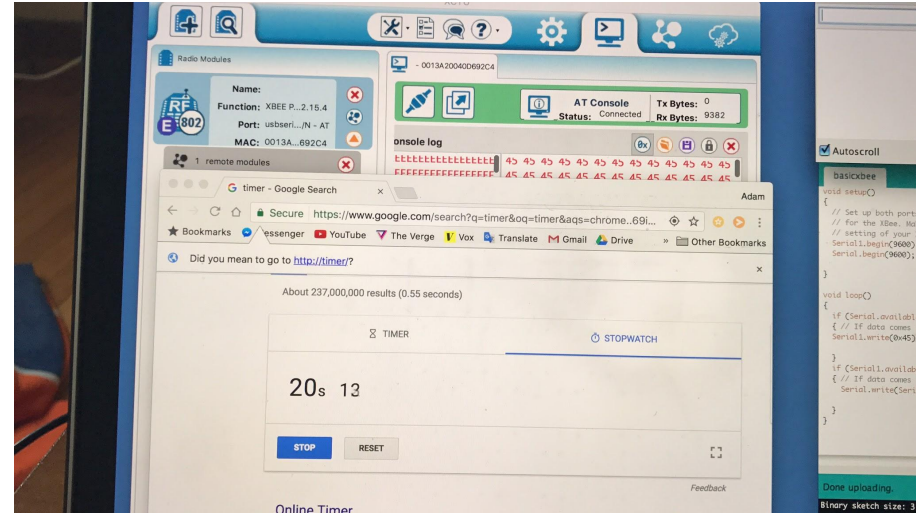
# Xbee Transceivers

- Xbee S1 chosen for ease of use with ATMega and adequate range and speed
- Like camera, main requirements on software that controls it

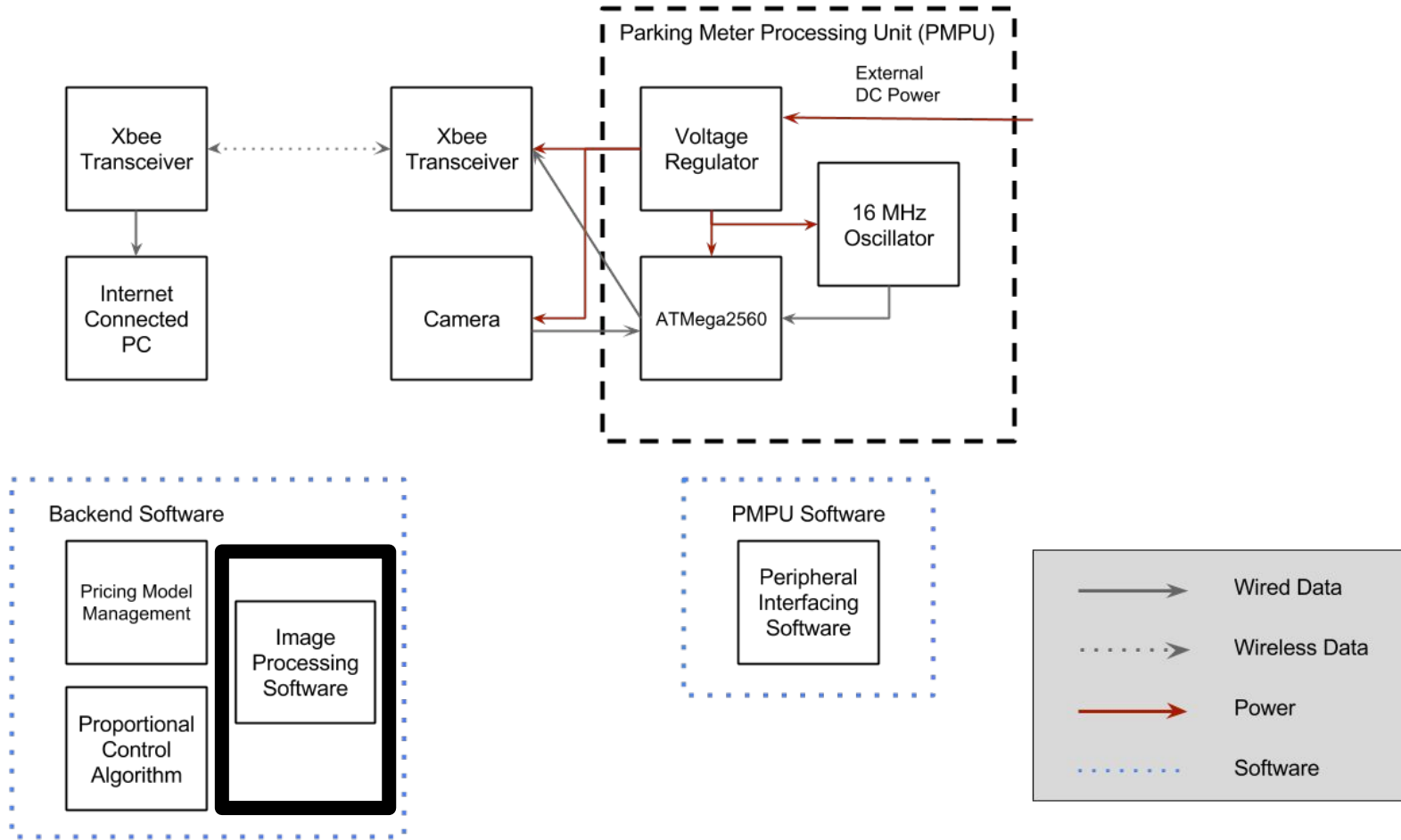


# Xbee Requirements and Verification

- Must be able to transmit between PMPU and computer at 1 kb/s
  - Shown at right: ~7.5 kb/s
- Must be able to transmit both directions up to 50 ft
  - Transmitted up to 400 ft during test



# Computer Vision Software



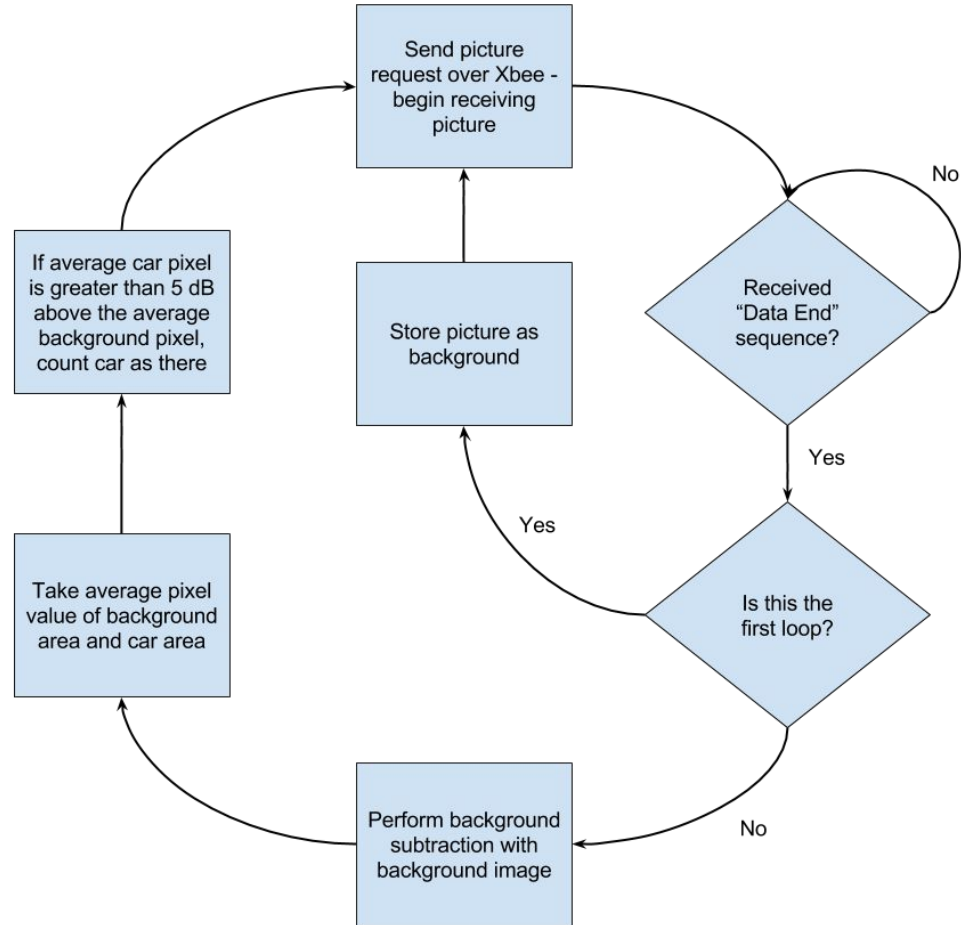
# Computer Vision Software Requirements

- Request picture from PMPU and store it within ~1.5 min
- Must be able to transmit data to PMPU and receive response within 30 seconds
- Must be able to perform Frame Differencing on a new image compared to a background image and attenuate the background by 5 dB
- Frame Differencing must run on PMPU

# Computer Vision Software Requirements

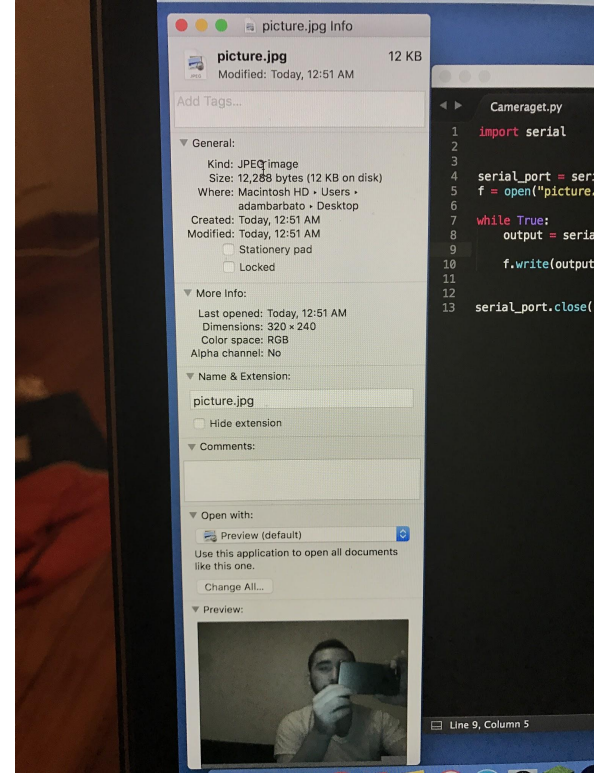
- Request picture from PMPU and store it within ~1.5 min
- Must be able to transmit data to PMPU and receive response within 30 seconds
- Must be able to perform Frame Differencing on a new image compared to a background image and attenuate the background by 5 dB
- ~~Frame Differencing must run on PMPU~~
  - Failed due to underestimating processing power and storage needed to work on two images within a microprocessor
  - Design going forward assumes it runs on backend





# Computer Vision Software Verifications

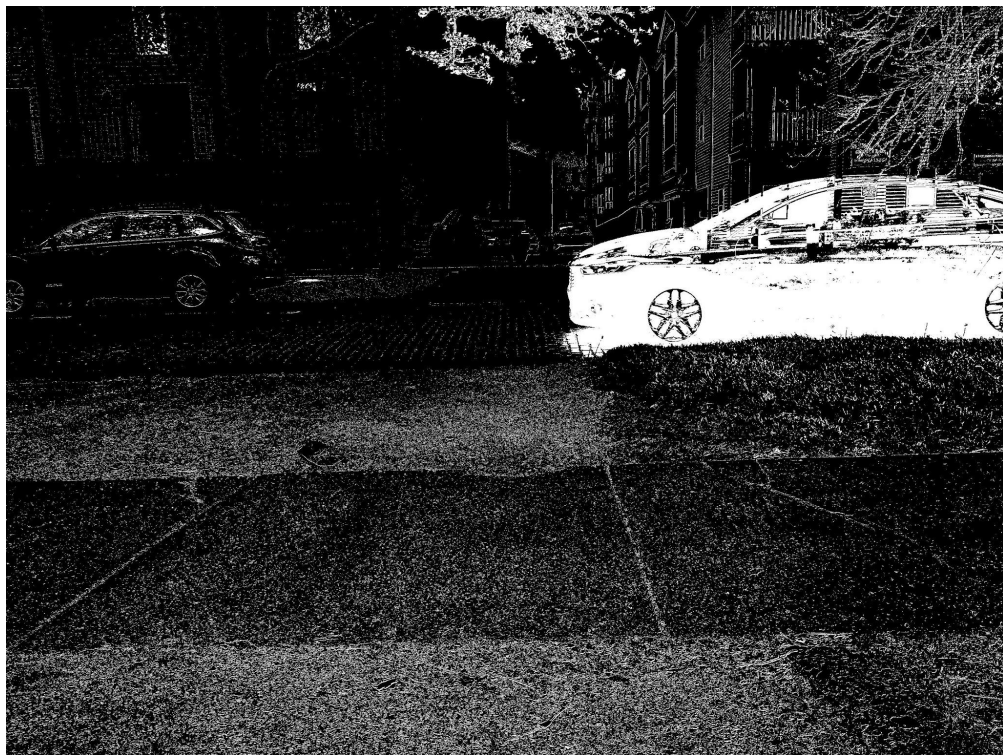
- Picture transmission shown in picture and video: ~20 sec for QVGA
- Response speed shown in video to be a few seconds



# CV Software Verifications - Frame Differencing

- Uses two pictures taken at two times
- One picture works as “background” that is stored at boot up
- Other picture is “current”
- Using a threshold, a sufficiently different pixel is marked white, otherwise black
- Then compare average car pixels to average background pixels









$\leq$  Car

$\leq$  Background



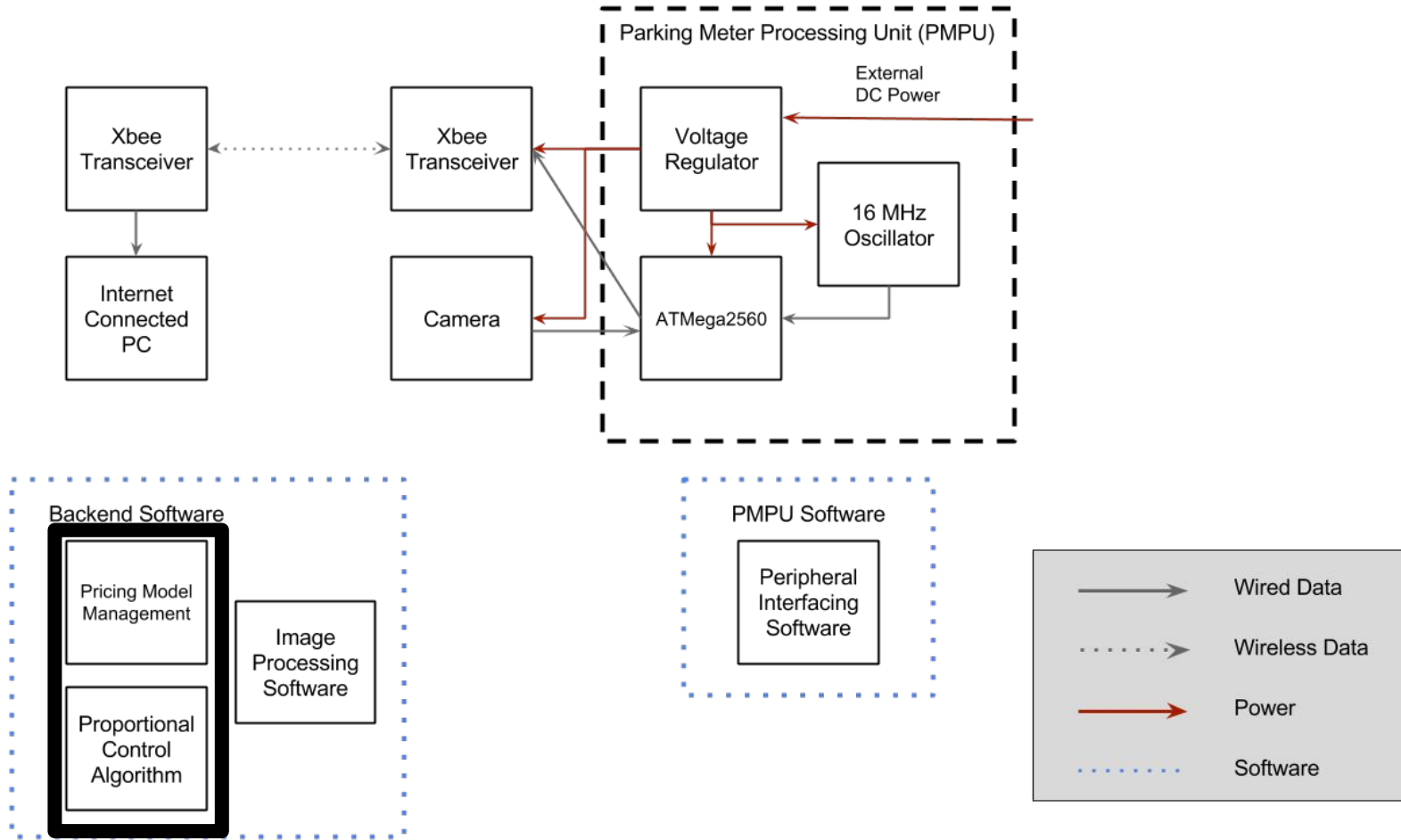
<= Car

<= Background

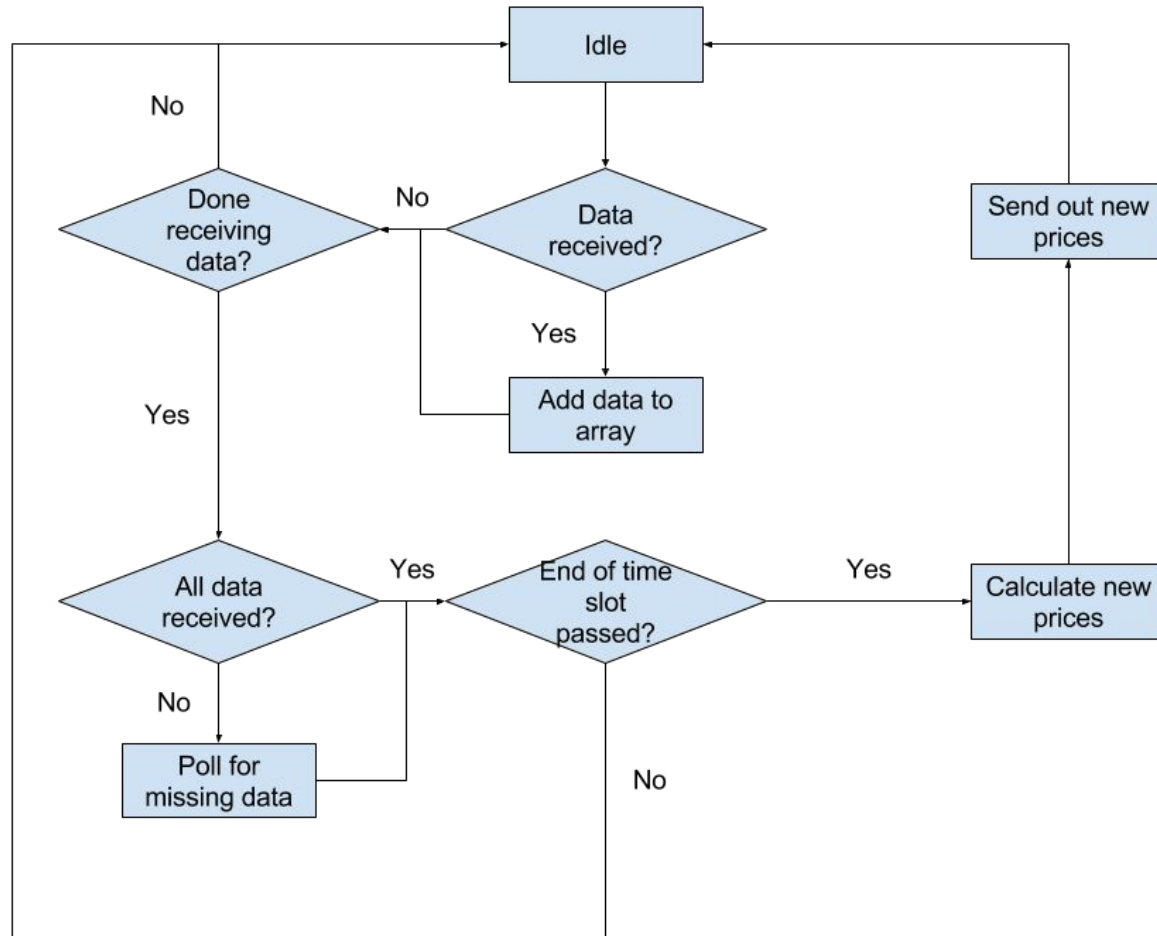
$$10 \log(\text{avgCar}/\text{avgBackground}) = 10 \log(156/19) = 21.05 \text{ dB}$$

# Backend Software





# Backend Software Flowchart



# Control Algorithm

Desired proportional change to prices based on local density comparison

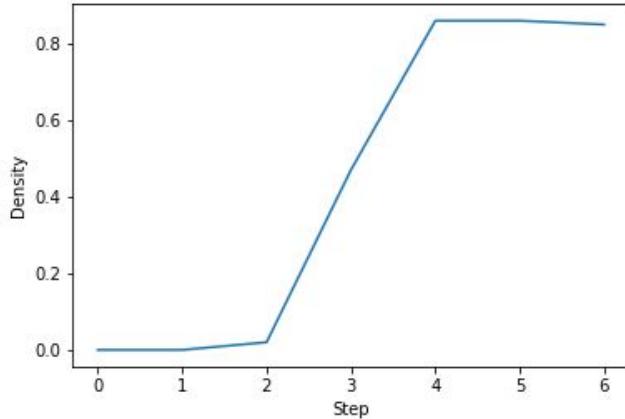
Algorithm prevents price from dropping below zero

# Control Algorithm

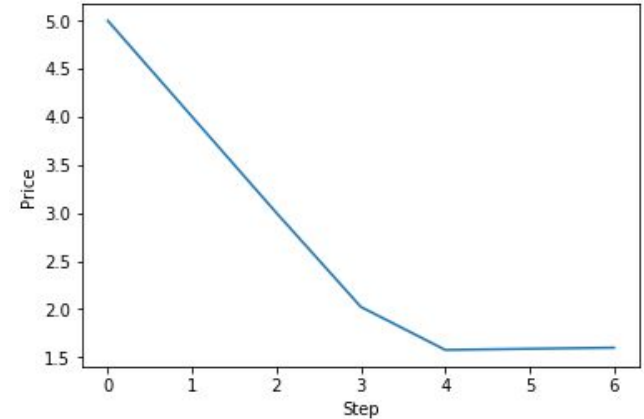
Single grid square test using 100 agents

Binary parking decision

Density Change



Price Change



# Simulation Test

Simulate the algorithm working over a larger space

10x10 Grid with 5000 agents

Agents have desired price and search radius

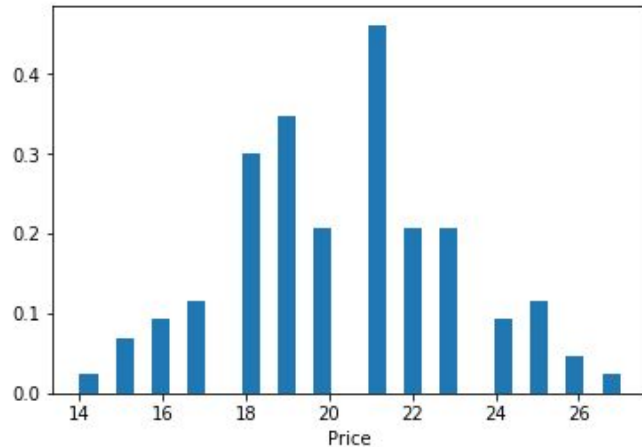
Run the control algorithm for 100 iterations, at each iteration all agents will attempt to park in a desired location

# Simulation Test

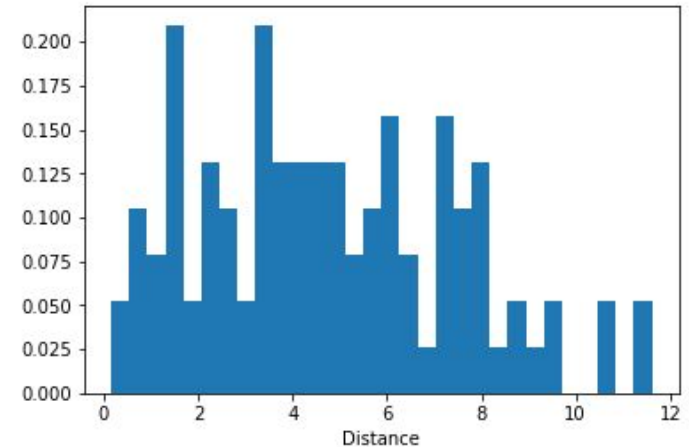
Personalities generated using normal distributions

Discrete for prices, continuous for distances

Desired Price (Dollars)



Search Radius (Grid Squares)



# Simulation Test

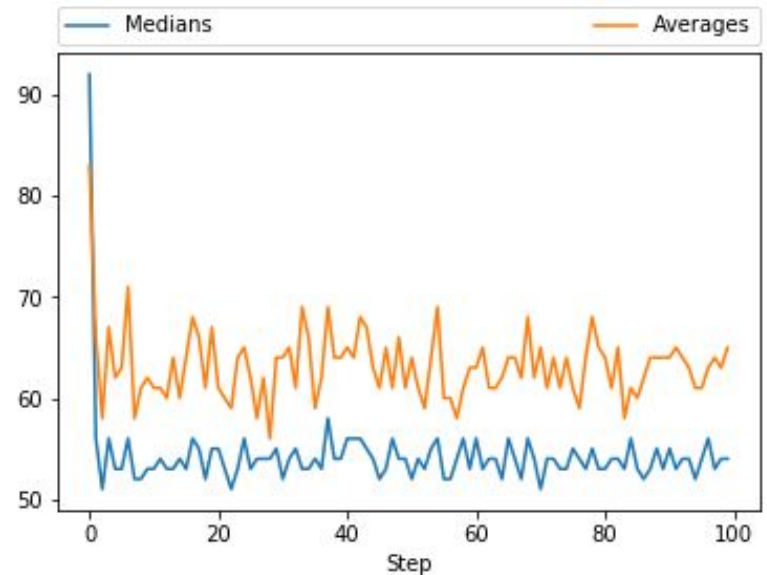
Graphed is the density over 100 iterations

Marginally Stable System

Non-zero average density  $\sim 65 \pm 5\%$

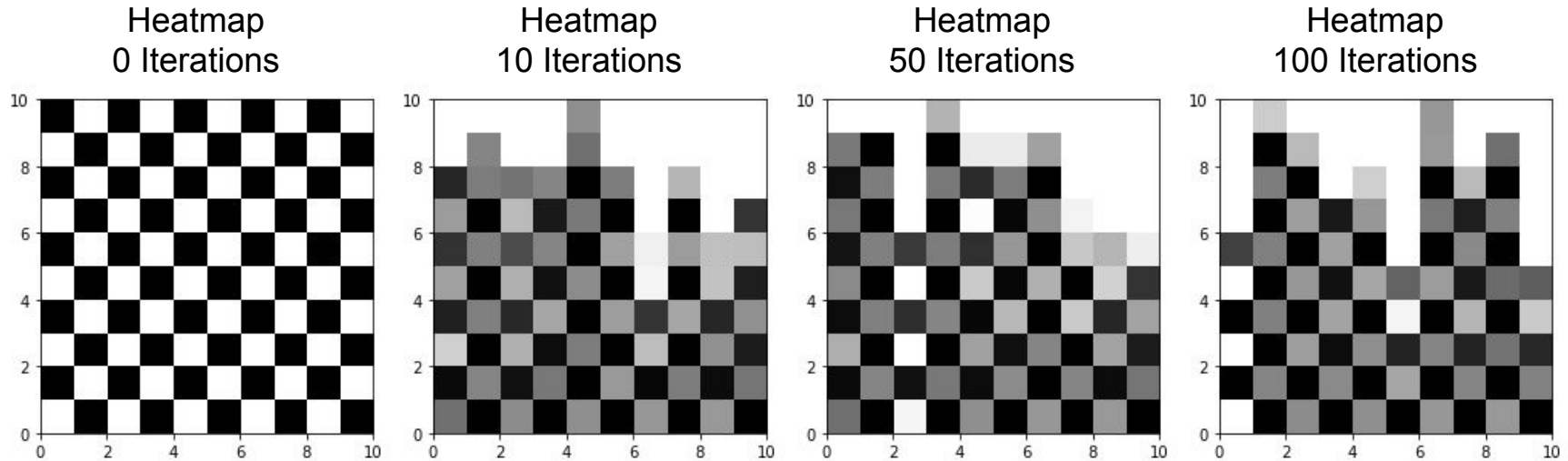
Median  $\sim 53 \pm 3\%$

Density at Each Iteration



# Simulation Test

Darker tiles indicate higher density





# Ethics

Like any project that manipulates prices, there is the opportunity for abuse by overseers

We believe our project, when used as intended is fair and ethical

# Future Work

In the future, the computer vision algorithm could be improved upon to create a more robust detection system, as right now it only works during specific circumstances

The control algorithm would require sufficiently more research to determine an appropriate solution for interacting with humans

# Conclusion

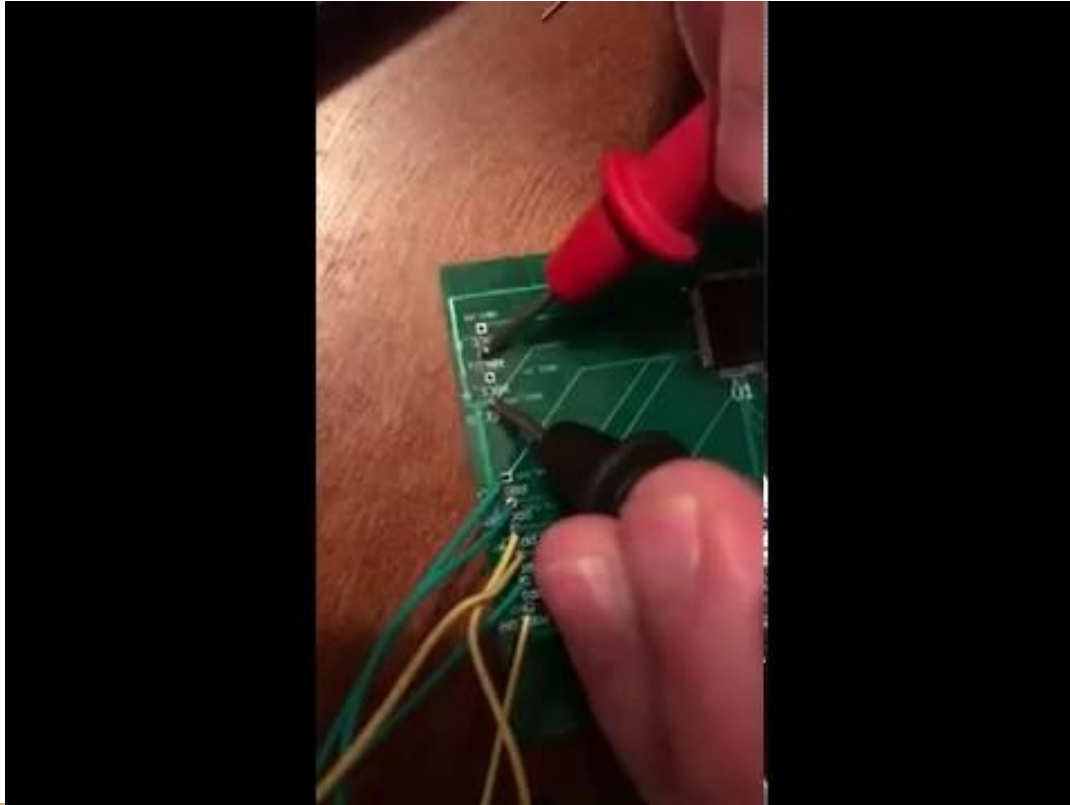
Simulation converged rapidly and became marginally stable, likely due to binary decision making of the agents

Computer vision algorithm correctly identifies the number of cars in the camera's field of view within administered constraints

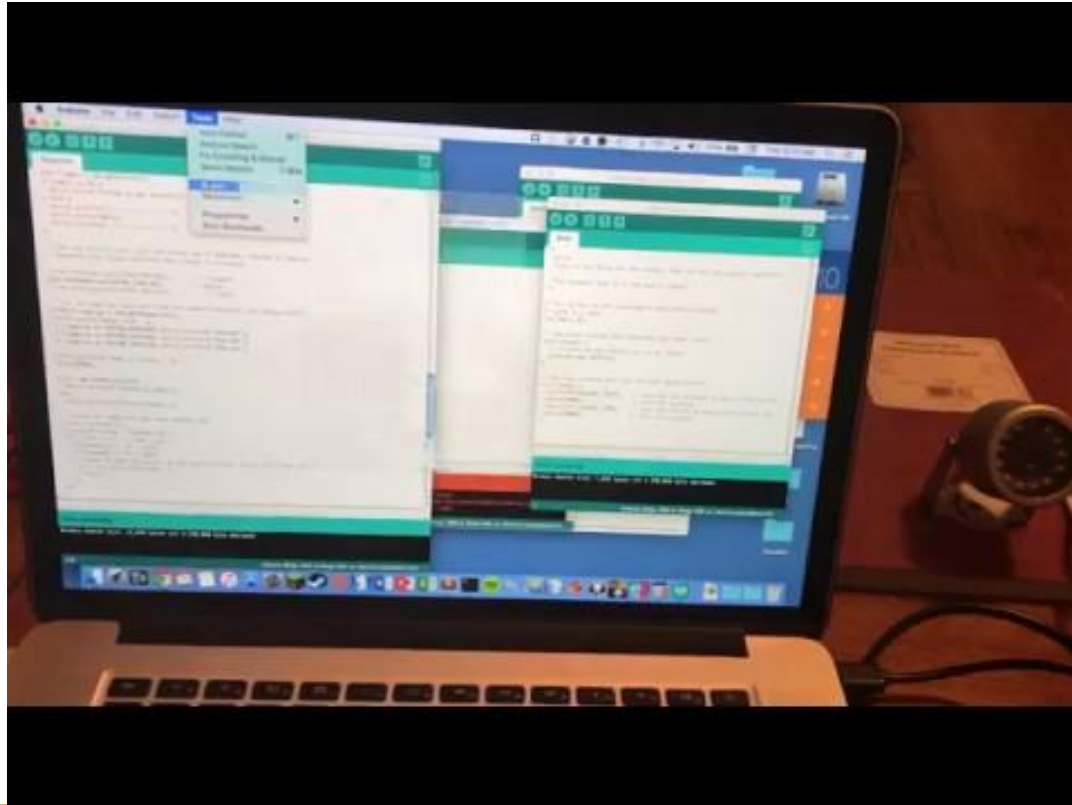
Hardware works as expected and gives a sufficient level of performance

# Supplemental Material

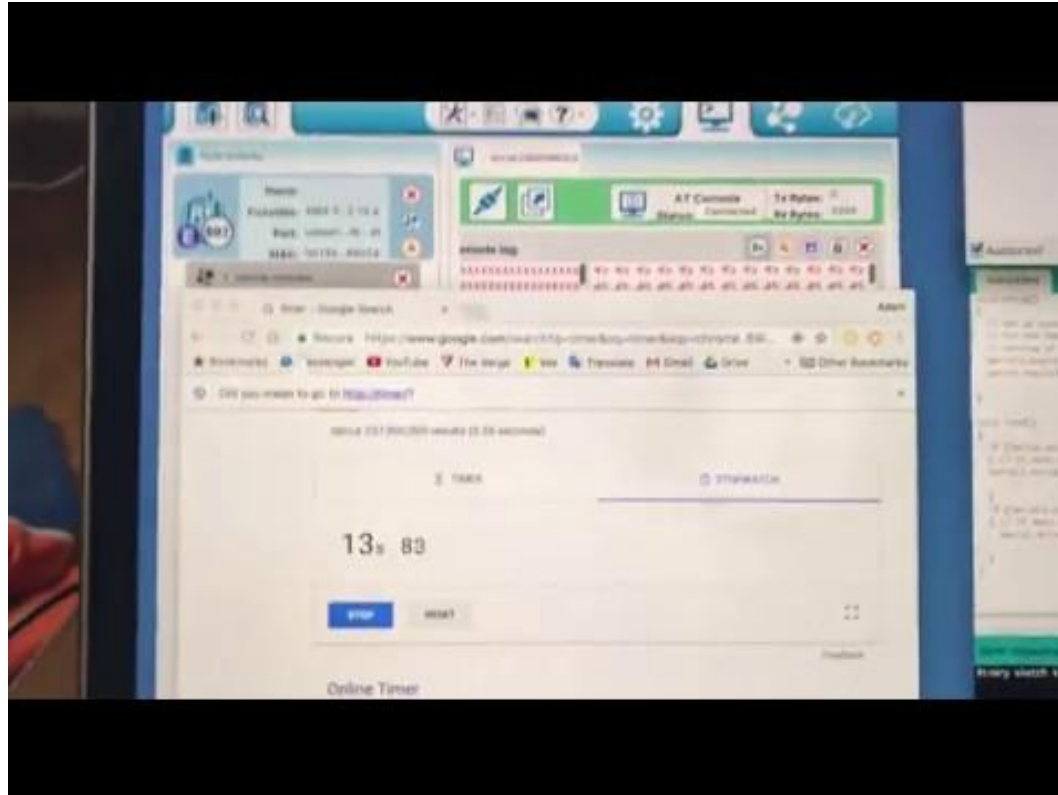
# PMPU “Blink” Test



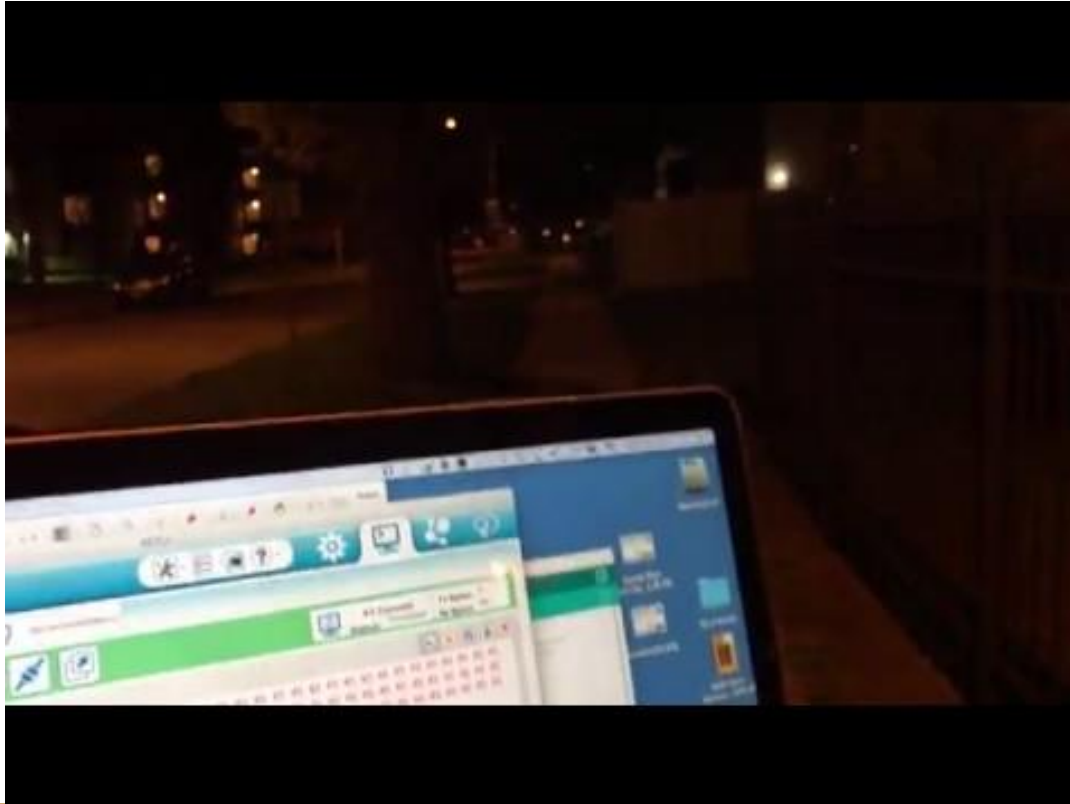
# Camera Picture Transmission and Storage



# Xbee Speed

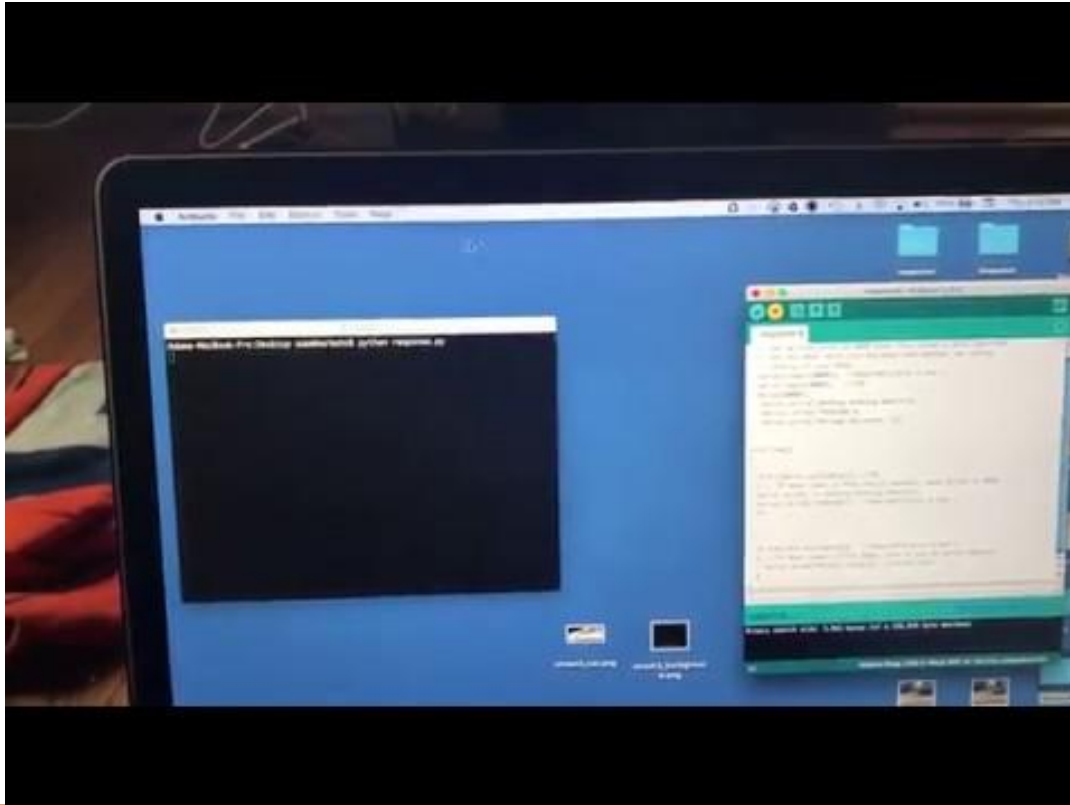


# Xbee Distance





# CV Software Response Time



P

- F

