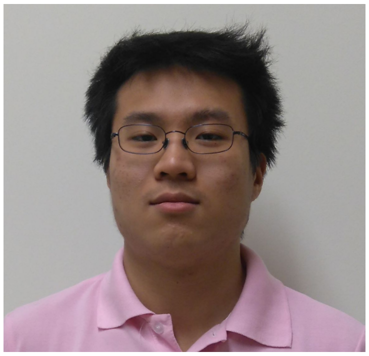
Wireless Autonomous Trolley

A robot that can deliver an object to a wirelessly specified location.

Final Report

Randy Baron

Philip Hoang



Abstract

The idea behind the Wireless Autonomous Trolley (WAT for short) is a convenient autonomous robot that can deliver objects to and from points specified by the user via a web page interface hosted by the WAT itself. The web page can be accessed from any device with WiFi capability and a web browser, and will allow the user to choose start and end points. The robot will navigate using a camera and image processing algorithms in order to find the marked destinations and carry items in a tray on top of itself.

This project is a low-cost implementation of this idea, but commercial versions that already exist are being used for real world applications such as delivering food in restaurants, hotels, at home, or at health facilities.

We have successfully implemented a functional prototype of the WAT. From parts that we could afford, we built a functional robot that uses image processing to seek out destinations specified by a user via WiFi.

Promotional Video:
https://www.youtube.com/watch?v=VQrq3w7PapU

Backup:
https://drive.google.com/file/d/0BwmGIH1Z9fbPZWVMMjILQzhsRW8/view?usp=sharing

Functional Requirements

The WAT contains a dedicated WiFi chip that acts as a server and hosts a basic HTML web page for the user to interact with. With this interface, the user can control the WAT in one of its two modes, autonomous mode or remote-controlled (RC) mode. In autonomous mode, the user instructs the robot where to go first (to pick up a load), where to go second, and then where the final destination is (where a human can pick up the load). The WAT navigates to these points of interest using a camera and image processing algorithm that runs on the WAT's main board (DE0 Nano). For the image processing to work, the points of interest must be marked by pre-placed objects. For our demonstrations, the pre-placed objects we are using are balloons of different colour. For practical use indoors, points of interest could be marked with coloured paper or printouts placed flat on walls or doors. In remote-controlled mode, the user can control the movement of the WAT using discrete commands to move it forward, back, turn left, turn right, or stop. The WAT also has an infrared proximity sensor in front to trigger an emergency stop if something unexpected blocks the its path.

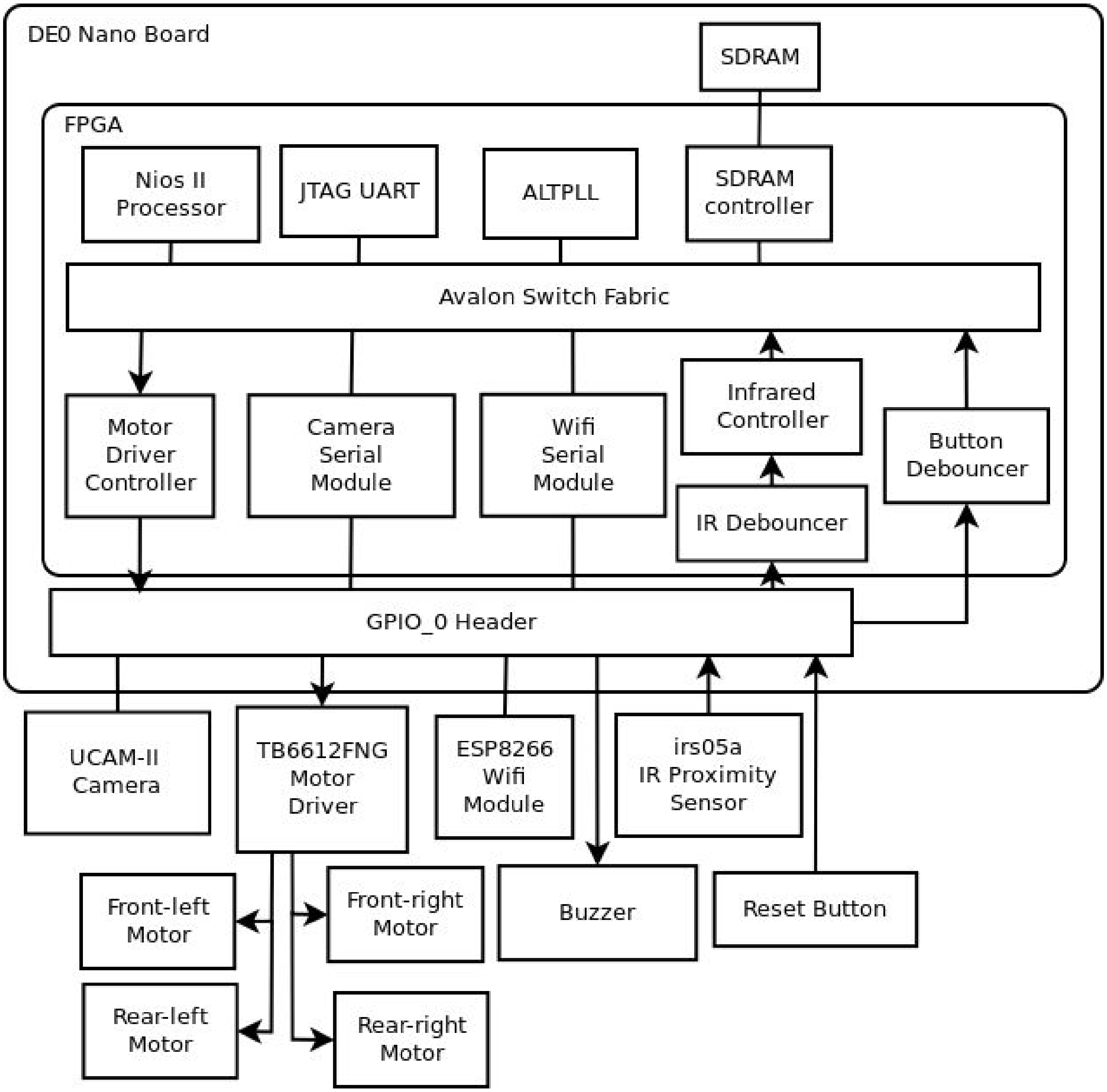
The purpose of the WAT is to autonomously move objects but one concern that was brought to our attention was weight capacity. Due to budget limitations, we ordered a relatively inexpensive chassis kit. The motors it came with are low power units. The result of this is that the WAT is only capable of comfortably carrying between 1 and 1.5 standard sized (355mL) cans of liquid. With 2 cans, there is enough power to move forward and back, but not enough to turn the robot. Putting these together we end up with an approximate maximum load of about 500 grams or 1 lb.

Our prototype is able to seek wirelessly specified destinations using its camera and image processing and carry a load to said destinations. The prototype can also operate like a remote controlled vehicle when the user wants it to. There are always refinements that can be made, but in the end, we were able to fulfill our specifications.

It was originally planned for the WAT to connect to a WiFi network and communicate with the user interface over the internet. This would not be suitable for future demonstrations as we would need to pay for a domain name indefinitely. Using the WiFi module's IP address to connect to it over a network would not work either because it would be dynamically allocated by the router it is connecting to. We shifted to using the WAT as an access point so any user in the present or future can use the WAT by connecting to it directly using WiFi. This means keeping it off of the wider internet which adds the advantage of not needing sophisticated security.

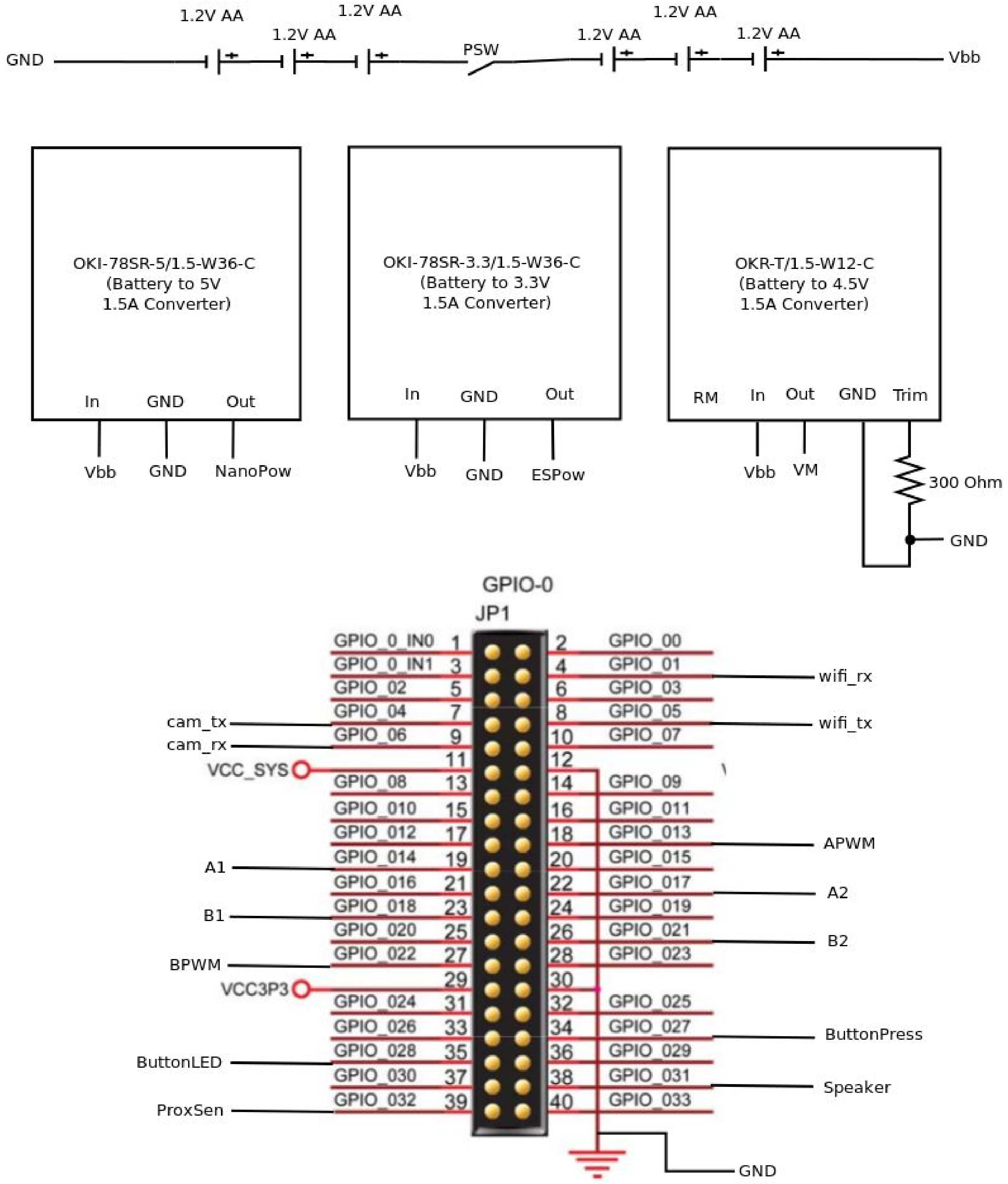
Design and Description of Operation

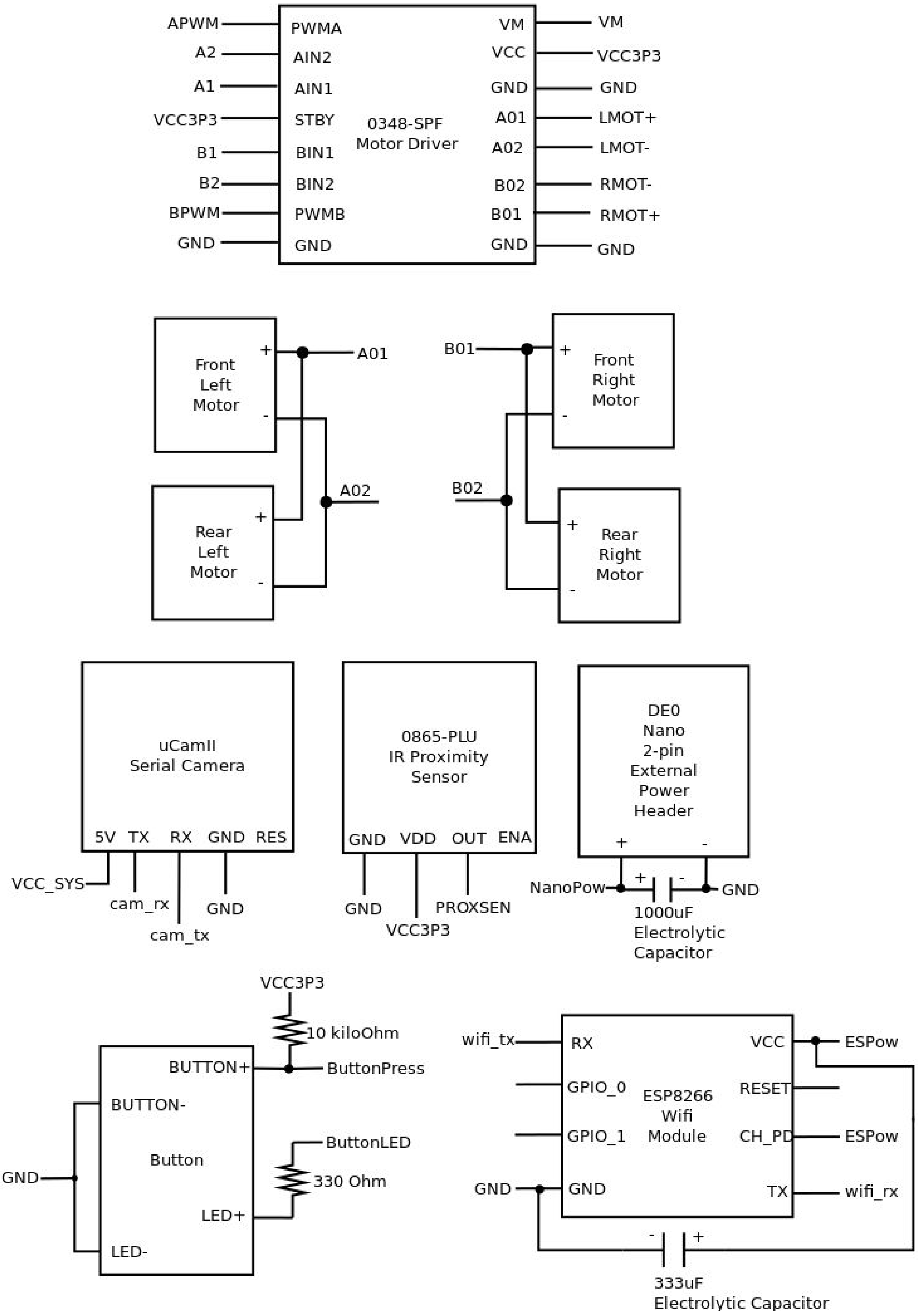
Hardware Block Diagram:



We chose to go with the TB6612FNG dual motor driver despite having 4 wheels for several reasons. First is because a previous ECE 492 group used it so we knew that it worked and we would have a reference. There is also no point in having one channel per wheel motor because this motor driver delivers ample current for every motor at full load and the action for steering would be exactly the same either way. Each channel is connected to 2 motors in parallel. The reset button and IR sensor go into debouncers which debounce the signals and deal with motor noise.

Comprehensive Schematic:




APWM
A2
A1
VCC3P3
B1
B2
BPWM
GND
PWMA
AIN2
AIN1
STBY
BIN1
BIN2
PWMB
GND
0348-SPF
Motor Driver
VM
VCC
GND
A01
A02
B02
B01
GND
VM
VCC3P3
GND
LMOT+
LMOT-
RMOT-
RMOT+
GND
Front
Left
Motor
+
-
A01
Rear
Left
Motor
+
-
A02
B01
Front
Right
Motor
+
-
B02
Rear
Right
Motor
+
-
uCamII
Serial Camera
5V
TX
RX
GND
RES
VCC_SYS
cam_rx
cam_tx
GND
0865-PLU
IR Proximity
Sensor
GND
VDD
OUT
ENA
GND
VCC3P3
PROXSEN
DE0
Nano
2-pin
External
Power
Header
+
-
NanoPow
+
-
GND
1000uF
Electrolytic
Capacitor
VCC3P3
10 kiloOhm
BUTTON+
ButtonPress
BUTTON-
GND
Button
ButtonLED
330 Ohm
LED+
LED-
wifi_tx
RX
GPIO_0
GPIO_1
GND
GND
ESP8266
Wifi
Module
VCC
RESET
CH_PD
TX
ESPow
ESPow
wifi_rx
-
+
333uF
Electrolytic Capacitor

We chose to use DCDC converters (voltage regulators) for power to have flexibility in terms of battery use. The regulators have a large input range so if for any reason we need to switch to another battery setup, we can simply plug it into the circuit without any modification to the circuit. With the regulator setup we have, the circuit requires between 7V and 15V to function properly. This means the WAT can use 7.2V, 7.4V, 8.4V, 9.6V, 10.8V, 11.1V, and 14.8V batteries commonly used in RC and airsoft applications. We chose to use AA batteries because they are provided by the lab. In our setup, with 6 AA alkaline batteries (6 x 1.5V = 9V), or with 6 AA rechargeable batteries either fully-charged (6 x 1.4V = 8.4V) or nominally charged (6 x 1.2V = 7.2V), the circuit will operate.

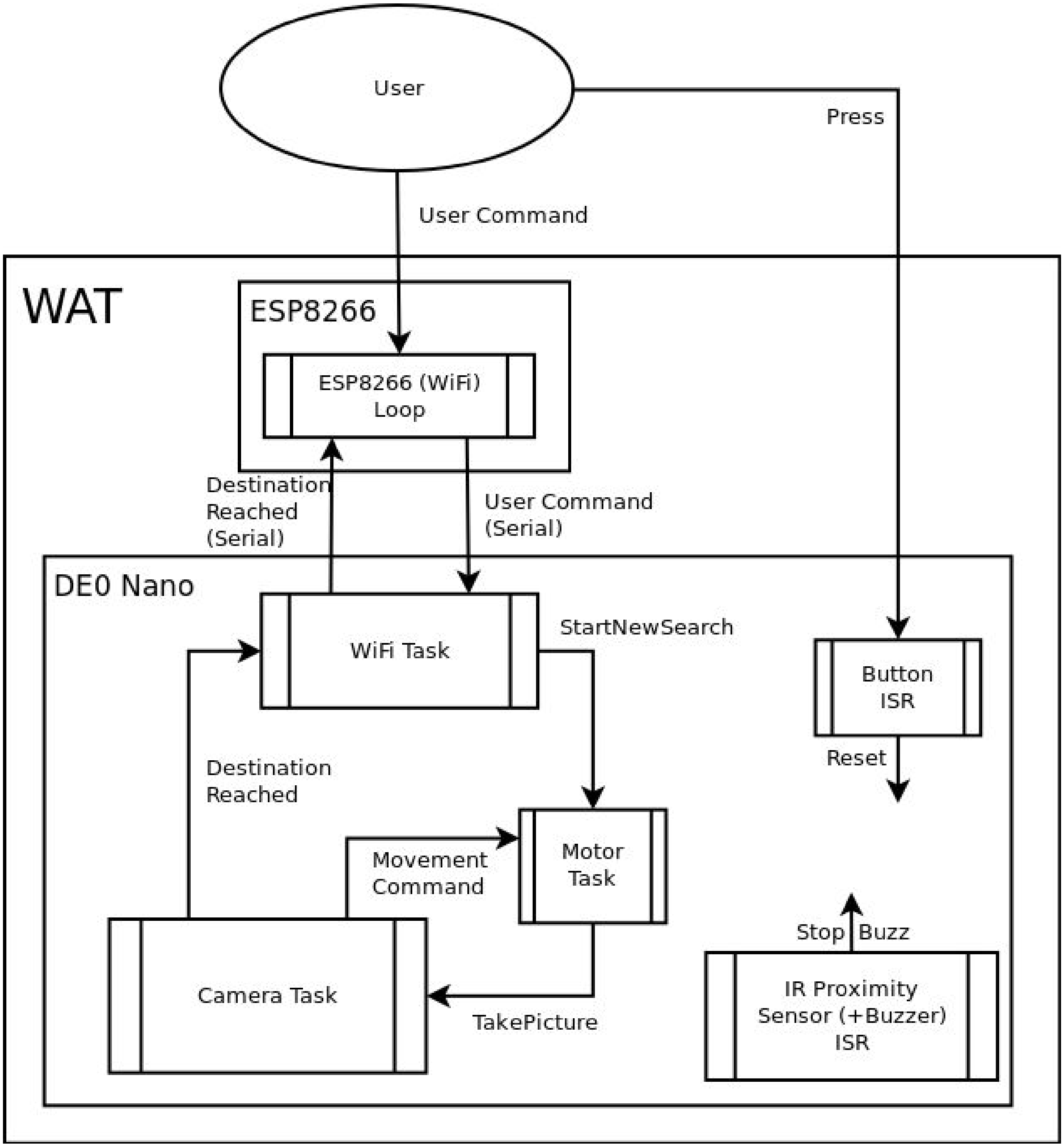
The regulator used for driving the motors has a variable output range of 0.591V to 6V. This is an advantage for us because the motors also have a variable operating range (4.5V to 6V). This means we can easily alter the the motor voltage to suit our needs in terms of torque, speed, and power consumption simply by changing the regulator’s trim resistor.

The ESP8266 WiFi module operates at 3.3V which the DE0 Nano can provide. However, the WiFi module can draw up to 300mA [10] which is more than the 3.3V rail of the DE0 can supply (200mA max), thus we needed to have a separate regulator for the WiFi module.

In the schematic, on the camera and WiFi module, rx signals are connected to tx pins and tx signals are connected to rx pins. This is because these signals are named in reference to the Nano, so the tx of the camera is the rx of the DE0’s camera component, the rx of the camera is the tx of the DE0’s camera component, etc.

At the power terminals of the DE0 Nano and the ESP8266, there are very large electrolytic capacitors connecting their VCC terminals to GND. It would be reasonable to think they are for motor noise suppression, but that is not their purpose. The reason capacitors are there is because when the motors spin-up or spin-down, they draw a large spike of current. Since the motors draw power through a DCDC converter that uses inductors, this current spike causes a massive voltage drop throughout the entire circuit which then causes the DE0 board and ESP8266 module to fail. This was an oversight during design, but the massive capacitors effectively counter the system failure by providing enough power to the DE0 and ESP8266 until the power distribution recovers. The capacitor sizes were found by running the robot and switching to bigger capacitors until it stopped failing.

Data Flow Diagram:



WiFi Loop:
Upon powering up, the ESP8266 starts its own WiFi network and acts as a server. Inside the ESP8266's loop, the ESP8266 waits for a client to connect and perform a request. It will then send the client an HTML file that the user can interact with. When the user sends a command using the web page interface, the ESP8266 reads the command and sends a corresponding command via serial to the DE0 Nano which is received and passed along by the WiFi task..

WiFi Task:
This task receives a command from the ESP8266's loop. If the command is an RC command, it will contain direction and magnitude information which motor task will parse. If it is an autonomous mode command, it contains information on which destinations to go to and time stopped at each. This information is passed along to the motor task though it basically just passes it along to the camera.

Motor Task:

1) Receives destination information from the WiFi task and keeps track of it.
2) Tells the camera task to take a picture alongside information of what to look for.
3) Receives movement information from the camera task after the captured image is processed and tells the motors what to do: turn, move forward, or stop.
4) **2)** and **3)** repeat until a destination is reached
5) Wait for new information from Wi-Fi task and repeat from **1)**

Camera Task:
Receives a command from the motor task to take a picture and process to look for a desired object. Once the object is located the position is turned into movement information for the motors and passed back to the motor task.

Button ISR:
Triggers a soft reset when the button is pressed.

IR Proximity Sensor ISR:
Causes the robot to stop if it is moving forward, and the buzzer to sound when the IR proximity sensor detects an object in front of it.

Bill of Materials

Part	Details	#	Supplier	Cost (CAD)	Link	Datasheet
Altera DE0 Nano FPGA board	⬦Programmable FPGA ⬦32MB of SDRAM ⬦1.445.15mA max draw (includes GPIO rails)	1	Lab supplied	$79.46	http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=165&No=593&PartNo=1	http://www6.in.tum.de/pub/Main/TeachingWs2016HSCDLegoCar/DE0_Nano_User_Manual_v1.9.pdf
Main chassis	⬦4 motors rated for 4.5V, 250mA max draw, 4.5W max power ⬦approx. internal dimensions: 4” x 7.4”	1	Canada Robotix	$38.29	http://www.canadarobotix.com/chassis-structures/1647-actobotics-half-pint-runt-rover-kit	http://www.robotshop.com/media/files/pdf/HalfPint_3View-637154.pdf
Motor Driver	⬦Max output of 3.2A, 1.2A average ⬦Standby control ⬦Filtering capacitors on both inputs	1	Canada Robotix	$11.59	http://www.canadarobotix.com/motor-drivers/sparkfun-driver-1a2-5v5	http://www.canadarobotix.com/datasheets/parts/TB6612FNG.pdf
Batteries	⬦6 AA 1.2V batteries in series = 7.2V overall	6	Lab supplied	$25.49		http://data.energizer.com/pdfs/nh15-2500.pdf
Camera	⬦UART interface ⬦DC 5V ⬦90mA draw	1	Digikey	$69.43	http://www.digikey.ca/product-detail/en/4d-systems-pty-ltd/UCAM-II/1613-1078-ND/5725466	http://www.4dsystems.com.au/productpages/uCAM-II/downloads/uCAM-II_datasheet_R_1_4.pdf
Infrared Proximity	⬦~30cm detection range	1	Canada Robotix	$7.69	http://www.canadarobotix.c	https://www.pololu.com/file/0J

Sensor	⋄IR emitter can be set on or off				om/infrared/pololu-38-khz-ir-proximity-sensor-fixed-gain-low-brightness	615/tssp77038.pdf
ESP8266 WiFi Module	⋄802.11 b/g/n ⋄Can act as host or receiver ⋄300mA max draw	1	Supplied by student	$7.69	http://www.canadarobotix.com/wi-fi/1788-esp8266-esp-01-serial-wi-fi-module	https://cdn.sparkfun.com/datasheets/Wireless/WiFi/ESP8266ModuleV1.pdf
Battery to 5V regulator (input to DE0)	⋄7V to 36V input range ⋄5V, 1.5A max current output	1	Digikey	$6.09	http://www.digikey.ca/product-detail/en/murata-power-solutions-inc/OKI-78SR-5-1.5-W36-C/811-2196-5-ND/2259781	http://power.murata.com/data/power/oki-78sr.pdf
Battery to 3.3V regulator (input to ESP8266)	⋄7V to 36V input range ⋄3.3V, 1.5A max current output	1	Digikey	$6.09	http://www.digikey.ca/product-detail/en/murata-power-solutions-inc/OKI-78SR-3.3-1.5-W36H-C/811-3014-ND/4878851	http://power.murata.com/data/power/oki-78sr.pdf
Battery to ~4.5V (variable) regulator (to power the drive motors)	⋄4.5V to 15V input range ⋄0.591 to 6V variable output range ⋄1.5A max current ⋄Require a 302Ω trim resistor	1	Digikey	$5.63	http://www.digikey.ca/product-detail/en/murata-power-solutions-inc/OKR-T-1.5-W12-C/811-2782-ND/3674288	http://power.murata.com/data/power/okr-t1-5-w12.pdf
10kΩ resistor	⋄Pullup resistor for button	1	Lab supplied	$0.007		

330Ω resistor	⬦Series resistor for button’s LED	1	Lab supplied	$0.007		
302Ω resistor	⬦Trim resistor for variable regulator	1	Lab supplied	$0.007		
333uF electrolytic capacitor	⬦Power cache for ESP8266	1	Lab supplied	$0.25		
1000uF electrolytic capacitor	⬦Power cache for DE0 Nano	1	Lab supplied	$0.28		
Buzzer	⬦Rated for 3V to 5V, 35mA max draw ⬦85dB at 10cm	1	Digikey	$1.46	http://www.digikey.ca/product-detail/en/cui-inc/CEM-1203(42)/102-1153-ND/412412	http://www.cui.com/product/resource/cem-1203-42-.pdf
Reset Button	⬦Has red LED built-in	1	Canada Robotix	$1.99	http://www.canadarobotix.com/mechanical-switches/1272-10mm-illuminated-pushbutton-red-momentary	
Power Switch	⬦Rated to 5A	1	Lab supplied	$1.99	http://www.canadarobotix.com/mechanical-switches/toggle-switch-3-pin-spdt-5a	
Plastic Bucket	⬦Attached to top of robot to carry load	1	Dollar Store	$2		
Rear bumper	⬦A foam bicycle handle	1	Dollar Store	$1.50		
Battery holders	⬦Hold 3 AA batteries each	2	Lab supplied	$1.00		

Wires and related	⋄includes ribbon cable, headers, crimps, solder	~	Lab supplied	$8.00		
Mounting gear and related	⋄includes screws, sticky tac, hot glue, velcro, sockets	~	Lab supplied	$5.00		

Total Cost: $280.94
Total Cost excluding provided equipment: $152.76
All prices are in Canadian dollars, shipping and tax not included

Available Sources

We used many of Altera's provided components on the DE0 Nano. These components are Altera's Nios II FPGA core [2], their Avalon ALTPPL core [1], JTAG UART core [3], SDRAM Controller Core [4], UART Core [5], Parallel IO [6], Timer core [7], and EPCS Core Flash Controller [8]. The basic UART Core provided by Alterra only has a 2 byte buffer which was a major bottleneck for our camera which needs to transmit images that can vary anywhere from 10-30kB in size. Fortunately an open source project exists that provides a FIFO UART [13] with an adjustable buffer size that solved our transmission problems.

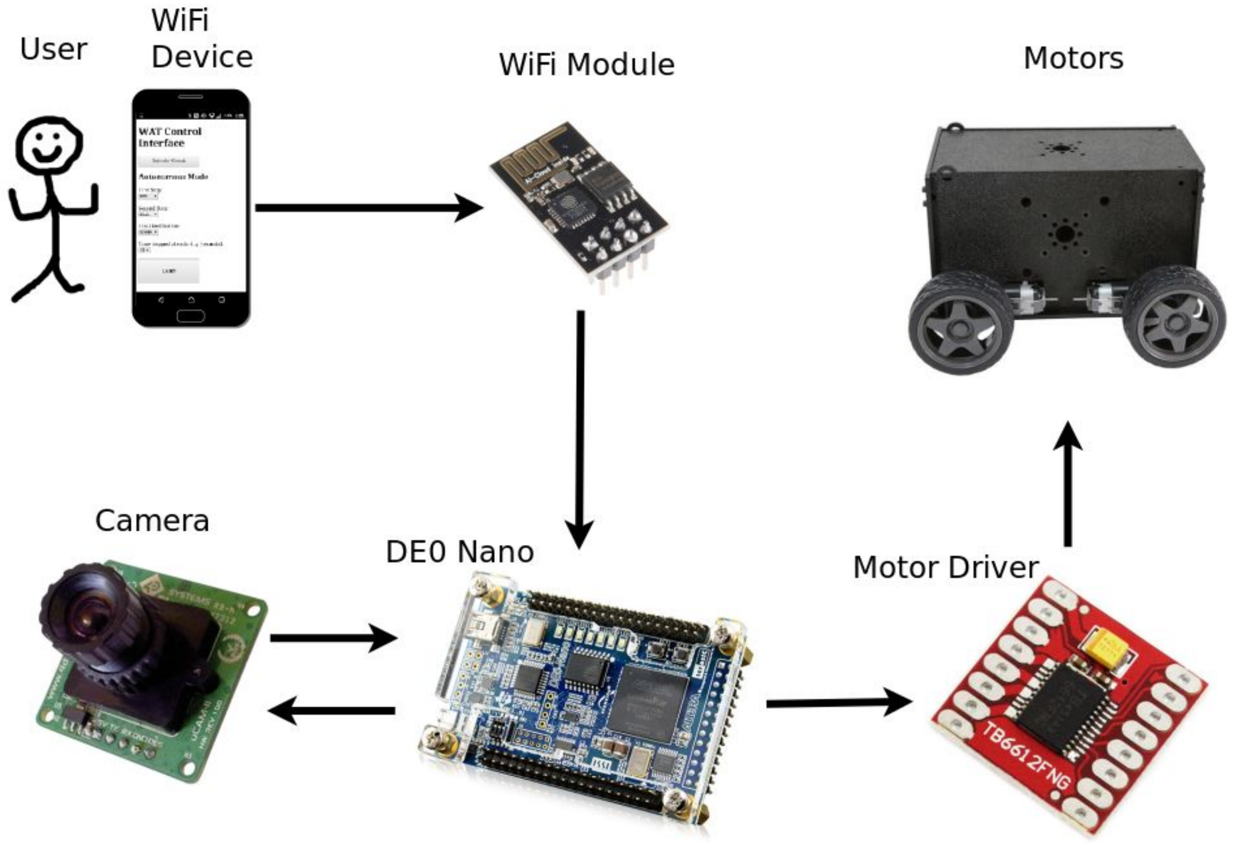
The Nios II full core will provide enough DMIPS (45 DMIPS at 50MHz [2]) for our application.

A key component of the image processing algorithm is an open source library called libjpeg, published by the Independent JPEG Group [12] which is required to decompress the JPEG image into a format we can collect data from.

The compiled .ELF file for the current project is approximately 2.0MB with the source files we've written being about 53kB and libjpeg library taking up about 1.3MB.

Datasheet

User-perspective block diagram:



Operating Conditions:

Battery Input Voltage	7 - 15V
Idle-state Current Draw	220 mA
Camera-state Current Draw	340 mA
Calculating-state Current Draw	280 mA
Moving-state (unloaded) Current Draw	570 mA
Moving-state (fully loaded) Current Draw	1040 mA
Maximum Current Draw (short spikes)	1270 mA
Operating Temperature	15 - 60 degrees celsius

Performance:

Maximum velocity (unloaded)	60 cm/s
Maximum recommended load	500 g
WiFi range	40 ft

IO Signals

Type	Name	Origin	Pin #	Destination	Pin #
Serial	cam_rx	Camera	2 (tx)	DEO GPIO_0	7
Serial	cam_tx	DEO GPIO_0	5	Camera	3 (rx)
Serial	wifi_rx	ESP8266	2 (tx)	DEO GPIO_0	2
Serial	wifi_tx	DEO GPIO_0	6	ESP8266	7 (rx)
Digital	ledg	DEO LEDs	0 - 7	N/A	N/A
Digital	motor_r_pwm	DEO GPIO_0	16	Motor driver	23
Digital	motor_r_a1	DEO GPIO_0	17	Motor driver	21
Digital	motor_r_a2	DEO GPIO_0	20	Motor driver	22
Digital	motor_l_pwm	DEO GPIO_0	25	Motor driver	15
Digital	motor_l_b1	DEO GPIO_0	21	Motor driver	17
Digital	motor_l_b2	DEO GPIO_0	24	Motor driver	16
Digital	prox_sensor	Proximity Sensor	2	DEO GPIO_0	37
Analog	Motor O A1	Motor Driver	1,2	Right motors	+
Analog	Motor O A2	Motor Driver	5,6	Right motors	-
Analog	Motor O B1	Motor Driver	11,12	Left motors	+
Analog	Motor O B2	Motor Driver	7,8	Left motors	-

Background Research

A similar project:
https://www.youtube.com/watch?v=YrOeHT2ZdDY

Using the ESP8266 as a web server and device controller:
http://randomnerdtutorials.com/esp8266-web-server-with-arduino-ide/

Real-world application:
https://www.youtube.com/watch?v=_Tdu2H1HHIA

Limitations of AA batteries:
http://www.powerstream.com/AA-tests.htm

Software Debouncing:
http://playground.arduino.cc/Learning/SoftwareDebounce

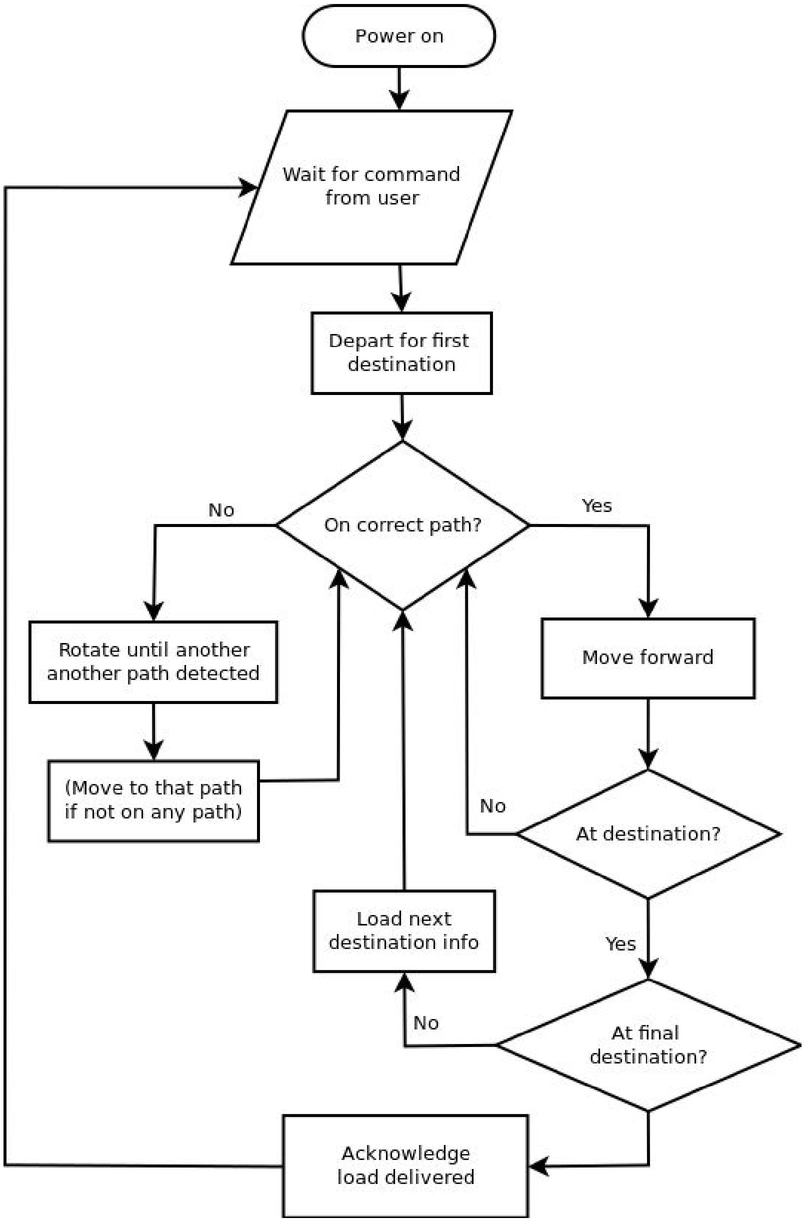
Electrical Noise from Motors:
http://motion.schneider-electric.com/support/general/detecting_shielding_noise.html

Electrical Noise Mitigation:
http://www.eetimes.com/document.asp?doc_id=1274125

FPGA implementation beats GPU implementation at image processing:
https://arxiv.org/pdf/1511.00100.pdf

FPGA acceleration for colour edge detection:
http://ieeexplore.ieee.org/document/6675272/
http://ieeexplore.ieee.org/document/6659389/

Software Design


Power on
Wait for command from user
Depart for first destination
On correct path?
No
Yes
Rotate until another another path detected
(Move to that path if not on any path)
Move forward
No
At destination?
Yes
Load next destination info
No
At final destination?
Acknowledge load delivered

Upon powering up the device will introduce three main tasks; Wi-Fi, motor, and camera. These tasks each handle one of three main aspects which are denoted by their name. A more detailed description of these can be found in the earlier design section.

Of note and not detailed there is the image processing algorithm the camera will be deploying. This makes use of the libjpeg library in order to decompress a loaded JPEG image into a struct containing all relevant information. The colour values for each pixel are analyzed and compared to one of two baseline ranges. The camera will automatically switch between a high luminosity and low luminosity capture depending on ambient light which means that we must check for two potential ranges. Any colour can be specified once appropriate range values are determined. Once a valid pixel is found for the colour region being searched for it will update internal values accordingly. Once the image is traversed the horizontal position of the region considered the most likely to contain the marker is turned into a value between 0-100 based on the distance away from the middle. This value is doubled and fed directly into the motor controls as a millisecond delay for the motors to be active. Under light load conditions this produces an almost ideal turn radius for zeroing in on a target. The robot will consider itself at a destination when it detects a certain amount of colour information, namely width, height, and pixels identified in the region.

There is a known memory problem with the robot unfortunately it emerged too late into development for us to try and find a way around it. We have ample resources but our problem is a fundamental one with malloc itself. In order to process an image we must first pass it into libjpeg which requires a FILE* and while this can be provided with native C functions these use the standard malloc function to allocate memory which are not strictly safe on an embedded system. We have 32MB of SDRAM available in total and a picture requires anywhere from 10-30kB to store. Our problem is that malloc looks for a single contiguous block of memory and ones of sufficient size become vanishingly rare on an embedded system creating an increasingly long bottleneck as image processing goes on. It's possible that the main spreading exists within the fairly expansive libjpeg which was also not designed with our system in mind and does a fair bit of memory allocation as well.

It is also possible though that this might not be a memory leak and more a matter of power leak. We noticed in the later stages of the design that the camera could become fickle if it doesn't get enough power, a possible source of this drain being the movement of the motors. An initial stage of operating the camera is to synchronize it with the DE0 and this requires an initial and increasing delay of 5ms so every failed synchronization attempt just further slows the board down as it takes increasingly longer between attempts.

Test Plan

Software:

Utilizing the hosts file system available through GDB debugging we are able to transfer files between the DE0 and a computer connected via USB. This allows us to get real colour thresholds and perform testing with a local C implementation running outside the board for performance reasons.

Hardware:

Pre-Assembly:

The motors and motor driver will be tested at the same time. The motor driver will be activated by the Wi-Fi module and send directional information to the motors.

The power supply of 6 AA batteries will be tested by measuring the voltage before and after a period of heavy use and comparing the results to typical voltage vs time curves. The camera will be tested by having it send an image to the SDRAM of the Nano or connected PC at different baud rates and compression ratios, and measuring the transfer time and quality.

The IR proximity sensor and button will be tested by hooking them up with power and reading the output pin with an oscilloscope to measure bounce time to determine appropriate detection amounts for the debounce VHDL circuit to correct. Since these are also interrupt driving devices a simple IRQ will be set up to ensure that we are achieving proper edge or level detection as needed.

The WiFi module will be tested by sending and receiving serial data to and from the DE0. The data to the DE0 will be printed to the connected PC's terminal, and the data to the WiFi will be printed on the web page. This will also provide directional instructions to the motor so we can ensure the control logic is appropriate for the motors.

The regulators will be tested by hooking them up to the a DC power supply and (afterwards) the battery array and checking their output voltages.

The buzzer will be tested by plugging it into the function generator. If everyone in the area becomes slightly annoyed, then the buzzer is working. Once this is verified we will be hooking it up to a timer so we can regularly annoy people with it.

Post-Assembly:

Motors and chassis capability will be tested by running the robot with varying amounts of weight.

IR sensor will be put to the test by driving the robot into various objects and observing whether the IR can detect it in time to prevent collision.

The WiFi module's range will be tested by sending commands at varying distances and behind different types of walls.

Results of Experiments and Characterization

Software:

Primary testing will be done on both personal laptops and the lab machine working from the assumption of 7.5 DMIPS for the DE0 and over 8000 DMIPS for a typical intel i7, a one thousand fold difference in performance. Early estimations of the image processing algorithm gave adjusted times of 200ms for a partially compressed 640x480 JPEG image. In reality this value varied between 3-8 seconds depending on how long the device has been running in autonomous. These times vary based on how long the device has been in operation but it's unclear if the slowdown is due to a power or performance bottleneck. Still, the speed is considered acceptable given the limitations of the platform we are working with. An unexpected result of the camera was the automatic luiminosity setting. There is a manual command to specifiy luminosity but the robot still decides to take a picture in either "light" or "dark" mode. This gives us two possible colour profiles we will need to check for and it's unclear which one we are working with. Acceptable thresholds have been collected for light or dark images based on images captured by the camera and sent over the hosts files system.

Hardware:

Pre-Assembly:

The motors were run with no load (other than the moment of inertia of the wheel) at operating voltage and were found to spin at ~30RPM. The current draw with no load was around 450mA, and at full lock-up was 1.27 A which is only just above the safety cut off of the motor driver at 1.2A and below the maximum current output of the voltage regulator of 1.5A.

The camera was found to be capturing 640x480 images that are approximately 10-30kB and since we are able to make use of a 921000kbps baud rate this image is transferred as fast as the CPU can process it. Since the camera also sends packets in sizes of 512 bytes the default UART device was considered incompatable and a suitable FIFIO Uart was found to fix this problem neatly.

The WiFi module was able to successfully provide a wireless network and host a web page that a client can connect to with any compatible device. Problems have emerged though with the motors providing significant power draw when changing directions.
The IR proximity sensor and button had long bounce times. When plugging them into the DE0 Nano to trigger interrupts, the interrupts fired multiple times when the proximity sensor activated or when the button was pressed. Because there is no real estate left on the perf board for a hardware debounce circuit, we used a software debouncing scheme and have our debounce managed to an acceptable level.
When the WAT is plugged into a DC power supply, all the regulators ran at expected voltages. The 5V regulator outputs 4.959V, the 3.3V regulator outputs 3.267V, and the 4.5V regulator outputs 4.580V.
The ESP8266 had a max current draw of 180mA (displayed on the DC generator) when transmitting.

The buzzer was plugged into a function generator and was found to operate at ~1-4kHz. Outside that range, there is a noticeable drop in volume.
The power switch's resistance from wire to wire was found to be 1.00 Ohms.

Current through the resistor for the LED (330 Ohm) was found to be less than 10mA which results in power dissipation of 0.011W (1.1V voltage drop across the resistor, 2.2V across the LED). The pull-up resistor of the button (10 kilo-Ohm) is pulled up to 3.3V so the current across it is 0.33mA which results in power dissipation of around 1mW. These are well within the operating bounds of the quarter-watt resistors we are using.

Post-Assembly:
The WAT's motors and chassis were able to move 2 standard sized cans (355ml each) forward and backwards, but did not have sufficient power to cause skidding to turn the robot. Running the robot with 1 can works completely fine. Due to the increased weight, turning and movement do show slow-down. We recommend the WAT carry no more than 500g of material.

The IR sensor was able to detect every object we placed directly in front of it. It detected these objects at varying distances, some objects getting detected at ~30cm and some at ~15cm. These are both acceptable distances for collision avoidance for our case. The IR only detects objects right in front of it, so it is unable to detect the metal legs of the chairs in the lab when they are not exactly in front of the sensor. Thus chairs legs and other thin objects are a hazard to the WAT. Since there is no sensor at the rear, we opted to install a foam bumper to protect the rear.

When doing endurance testing on the WAT, we found that it could run reliably for no more than 20 minutes continuously so spare batteries are recommended for demonstration. The WAT runs reliably with NiMH rechargeable AA batteries, but when we used disposable alkaline AA batteries, even when they were brand new (1.6V), they could not sufficiently power the robot and the camera would fail very quickly. Alkaline AA batteries work fine for RC mode however.

Safety

There is very little serious danger associated with this project;
Maximum Voltage: The batteries will provide up to ~8.4V and this will at most cause a imperceptible shock to humans. Voltage regulators are included to ensure components are kept at a safe operating voltage to prevent failure.
Stored Energy: The AA batteries each hold 2650mAH capable of discharging at 2C (5.3A).
Operating Temperature: All components can operate within a range of at least -20C to +60C and since this is intended to be operated indoors this should not be a problem capable of leading to component failure.
Velocity: Maximum unloaded velocity was found to be 0.6m/s.
Kinetic Energy: There is potential for this to be a problem if the robot is carrying a potentially dangerous load but the infrared distance sensor is being included to prevent collisions for exactly this reason. As for the robot itself, unloaded the robot weighs in at just under a kilogram which gives an expected maximum unloaded kinetic energy of 170mJ. Provided a load does not decrease the speed by less than 33% this value will not increase.
Potential Energy: Negligible, the robot will be on the ground during operation.

Regulatory and Society

Since there is no personal information of any kind being stored there is no relevant information privacy laws applicable to the project. There is however a risk of being hacked thanks to the wireless aspect of the design. The wireless module will only be active when the robot is active and a simple password encryption, different from the default ones, should be sufficient.This device does not connect to the internet in its current iteration, it is a purely local network operating with the WiFi protocol.

RF interference seems to be short range, contained and minute with the electric motors providing a large source of it. The Wi-Fi module follows IEEE 802.11 and operates at a maximum output of 20dBm, well below the FCC maximum of 30dBm.

Environmental Impact

The main hazardous substance in the WAT is lead from the solder. There are lead-free solders available.

The uCam-II, Infrared proximity sensor, buzzer, and voltage regulators are all labeled as ROHS compliant. The ROHS status of all other components is not stated by the manufacturer so we can assume that they are not compliant.

In addition to the potential lead solder, the WAT is non-biodegradable so must be disposed of properly at the end of its useful life.

Sustainability

Power consumption

We plugged the WAT into a DC power supply and observed current draw at a set voltage and multiplied this current and voltage to get power.

Idle state:

0.22 A at 7.2 V = 1.584 W

CPU under load state (calculating):

0.28A at 7.2 V = 2.016 W

WiFi max power state (transmit mode):

0.47 A at 7.2 V = 3.384 W

Camera max power state (taking picture and transmitting):

0.34 A at 7.2 V = 2.448 W

Moving state (motors spinning):

No load:	0.57 A at 7.2 V = 4.104 W
Max load:	1.04 A at 7.2 V = 7.488 W
Average:	5.796 W

When powered on and not running, power consumption would be 1.584 Watts.

When powered on and running at an estimated 60% of the time in the moving state, 20% in camera taking picture and transmitting state, and 20% in calculating state (negligible time spent in WiFi mode), power consumption would be 4.3704 Watts.

In this running state, if the robot ran non-stop, it would consume 38.31 kWh which translates to roughly $1.33 energy cost per year. If coal-based electricity is used, up to 37.79kg of C02 could be generated to power this robot per year [11].

It would take approximately 0.02 square meters of solar panels to continuously power the WAT.

References

[1] “ALTPPL (Phase-Locked Loop) IP Core User Guide.” [Online]. Available: https://www.altera.com/en_US/pdfs/literature/ug/ug_altpll.pdf

[2] “Nios II Processor.” [Online]. Available:
https://www.altera.com/products/processors/overview.html
https://www.altera.com/en_US/pdfs/literature/hb/nios2/n2cpu_nii51015.pdf

[3] “JTAG UART Core.” [Online]. Available: https://www.altera.com/zh_CN/pdfs/literature/hb/nios2/n2cpu_nii51009.pdf

[4] “SDRAM Controller Core.” [Online]. Available: https://www.altera.com.cn/content/dam/altera-www/global/zh_CN/pdfs/literature/hb/nios2/n2cpu_nii51005.pdf

[5] “UART with FIFO Buffer MegaCore Function User Guide.” [Online]. Available: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_uart_fifo.pdf

[6] “Parallel Port for Altera DE-Series Boards.” [Online]. Available: ftp://ftp.altera.com/up/pub/Altera_Material/9.1/University_Program_IP_Cores/Input_Output/Parallel_Port.pdf

[7] “Timer Core.” [Online]. Available: ftp://ftp.altera.com/up/pub/Altera_Material/9.1/University_Program_IP_Cores/Input_Output/Parallel_Port.pdf

[8] “EPCS Device Controller Core.” [Online]. Available: https://www.altera.com/zh_CN/pdfs/literature/hb/nios2/n2cpu_nii51012.pdf

[9] Scott Larson, “Debounce Logic Circuit (with VHDL example).” [Online]. Available: https://eewiki.net/pages/viewpage.action?pageId=4980758

[10] “ESP8266 Quick Start.” [Online]. Available: https://benlo.com/esp8266/esp8266QuickStart.html

[11] “US Energy Information Administration FAQ.” [Online]. Available: https://www.eia.gov/tools/faqs/faq.cfm?id=74&t=11

[12] libjpeg, Independent JPEG group Available: http://www.ijg.org/

[13] “FIFOed Avalon Uart.” [Online]. Available: http://alterawiki.com/wiki/FIFOed_Avalon_Uart

Quick Start Guide

Quick Start Video:
https://www.youtube.com/watch?v=JG3y5VckEVc

Backup:
https://drive.google.com/file/d/0BwmGIH1Z9fbPMS1rWnMwUENwbG8/view?usp=sharing

Starting the device up should be fairly simple assuming it is willing to cooperate with you. The latest and final code is flashed onto it currently so there should hopefully be no need to reprogram it. Should that need arise you can use the Nios II Flash programmer to quickly get it set back up. Ensure it is connected and you have the latest commit from the final branch of our github page and load WAT_MAIN.flash_settings into the flash programmer and hit program.

There are only two external physical peripherals that users can interact with; the power switch and the reset button. The OFF position of the power switch is pointing down and ON is pointing up. Ensure that batteries are fully charged otherwise you may encounter problems when trying to operate in autonomous mode. The reset button also doubles as a minor status light, as long as that is on the robot is in normal operating mode. If it goes off (or never turns on) that is a sign that you are likely having power problems or something went wrong internally and a hard reset is required.

Our most common solutions to this problem are in order:

- Between the robot and the battery pack is a switch. Ensure this is facing outward on both sides of the robot as these are often jiggled when changed batteries.
- Ensure batteries are properly mounted
- Ensure batteries are adequately charged
- Ensure batteries are at operating temperature (not fresh out of a fridge of charger)
- Ensure battery connectors inside are securely connected inside
- Ensure the capacitors are properly inserted into their sockets
- Ensure all other connectors are connected to their respective pins

If any of those above steps fail then there is probably something wrong with the robot that cannot be quickly or easily diagnosed.
Assuming you are able to get the robot operational the light will turn on and after a short delay (anywhere from 5-10 seconds) the wifi module will be booted up and providing a WiFi network. The credentials are:

Name: WATwork
Pass: ece492w2017
Page: 192.168.4.1

Enter that IP into your browser of choice from a wireless device of choice and the wireless interface will appear. This basic HTML page gives you access to RC and autonomous mode. For RC mode push the button and it will move accordingly, which should be straightforward. In Autonomous mode you will select three destinations that the robot will search for and in what order. These destinations must be marked by a coloured object at least 6 inches tall and wide. It supports red, green, and blue and while we used cheap balloons from a dollar store, it should be able to pick up on suitable coloured objects assuming you can find a good reference. Try the WAT's autonomous mode on a coloured object from different angles before doing a demonstration with it. Once it has homed in a destination it will play a short tone sequence to let you know it is moving onto the next marker.

There are two things to be aware of when operating the device:

- The IR sensor can be finicky. You'll note it's mounted on a small amount of sticky tac to keep it pointing up. It has a nasty habit of triggering off of the floor if you don't point it up far enough. You'll know if it has fallen too far if it beeps instead of going forward.
- If the rear light turns off during autonomous mode that is a likely indicator that the camera has crashed and a hard reset is required, potentially also a change of battery.

Future Work

If we were to build a commercial version, we would include internet capability as this would enable the user to use the WAT even when out of range of the WAT's WiFi network.

There are many aspects of the interface that could be expanded. One aspect we did not have time to implement was communication back to the user from the robot. This would be used to tell the user status of the robot like completion or internal failure. For demonstration purposes, the autonomous mode only has 3 possible destinations (red, blue, and green), but this could be expanded to use multiple colours and/or patterns of colours. In a future iteration, we would also include the ability to save routes so the user could send the WAT somewhere by selecting a previously inputted route instead of building it every time. Right now the web page offers an option to stop at each destination for a specified amount of time, but a stretch goal would be to have an option to stop at specified destinations until an ACK is received by the WAT. This ACK could be in the form of a physical button being pressed on the WAT, or a button on the web page interface itself.

Source code and hardware documentation

These documents can be found at https://github.com/heiga/WAT

There are three main groups of code;

- **Quartus, contains our Quartus project and all code associated with the running and testing of the robot, both C and VHDL.**
- **Esper; this is the code being flashed onto the WiFi module**
- **Localtest; this is what amounts to our software testbench. Nothing of substance really happens here and most code on here is more or less junk.**

Within Quartus is where you will find our top level design and supporting vhdl code.

- **ECE_492_g7_2017w_WAT.vhd - E**
  - **The top level of our code that specifies connections between external components and internal ones specified by QSys.**
- **Debounce.vhd - E**
  - **Simple debounce circuit that provides a VHDL solution to the problem. Once triggered it starts a counter and if the signal remains high for a long enough period it is considered to be logically “1”. This part has been integrated into QSys.**
- **Speakermodule.vhd - E**
  - **Provides a continuous square pulse of a given frequency so that the speaker can beep annoyingly with the application of a logical “1”. This part has been integrated into QSys.**

Within Quartus/software/WAT_MAIN is where our actual design lies as the name would suggest. This is what is flashed onto the board.

- **Camera - E**
  - **Controls the ucam-ii camera, telling it when and how to take a picture.**
- **Imagepro.c - E**
  - **Contains our core image processing algorithm that the camera will use to collect data about the picture it just took.**
- **Motor - E**

    - Controls the motors based on values given to it by the other two main tasks.
- Wat_main - E
    - Contains the main function and initializes all tasks, interrupts, and queues. Performs no code beyond these initial steps.
- Wifi - E
    - Controls the wifi module and passes control information along to motor.

Within Quartus/software/WAT_TEST is the testbench used for ensuring IO. As with localtest the state of this code should be taken at face value. We finished hardware testing long ago but we might occasionally use this to test something without disrupting the main project.

- Camera_test - E
    - Tests the camera functionality and ensures it works properly
- Motor_test - T
    - Tests the motors though this is currently an empty loop as wifi took over that task once we knew we had the basic command figured out
- Wat_main_test - C
    - Contains the main function and initializes all tasks, interrupts, and queues
- Wifi_test - T
    - Controls the wifi module, and as of the most recent iteration, the motors as well. Accepts a code from the user and turns that into directions for the motors.

Within Esper is the flash code that actually controls the wifi module

- EsperComplete.ino - E
    - Contains the operational details for the UART interface and is where the HTML code that generates the displayed code is stored as a literal string.

Within localtest/src is the current iteration of software not yet integrated to the Nios 2 system or just being used for testing purposes.

- Imagepro.c - T
    - This was named as such since it was the original location of the image processing algorithm while it was being initially developed. It

is still used primarily for testing image processing but other things may occasionaly end up included.