ECE 445 Design Review

A Parallelized Algorithm for Hyperspectral Biometrics

Christopher Baker

Timothée Bouhour

Akshay Malik

Table of Contents **Page No**

I. Introduction

Overview / Motivation

In the modern world unique person identification has become an increasing challenge, central to strategies in combating terrorism and crime to provide global security. Recent research has shown that hyperspectral imaging provides new and improved biometric data, which can be leveraged to meet this challenge by examining features in different spectral bands.

Despite its promise, this method of identification still has some challenges, which must be addressed before it can be applied in the real world. One of those challenges is dealing with the massive amount of data that a hyperspectral sensor generates. We will be developing an algorithm to solve this data processing problem based on GPU parallel processing. Our algorithm will be scalable to allow it to be expanded and used as the technology develops into real world application.

Objectives

Our overarching goal in this project is to provide a high throughput processing back end for biometric identification based on hyperspectral images of a person's face. Our product will be an algorithm coded and running on a GPU using the CUDA C programming language. Our specific objectives include the following:

1. Determine which facial features work best for hyperspectral biometrics
2. Develop, code, and test an optimized parallel identification algorithm

Functions

We aim to provide a solution such that any hyperspectral camera can be used to collect data. Once the data is formatted properly, we should then be able to identify the person in the picture given that we have previous hyperspectral data from that person. The functions of our final product include the following:

1. Identify a person against a database based on a hyperspectral image of their face
2. Match a hyperspectral image of a face against a database using a GPU
3. Efficiently store and access a database of hyperspectral images
4. Store various access levels for each individual in the database, and spin a certain amount of motors to emulate the opening of doors based on access level when a match is identified.

Benefits

Using our product will provide the customer with the following benefits:

1. Provide a better and more reliable way to recognize individuals
2. Provide processing for massive facial data input in near real time

3. Identify a face despite changes (mask, aging, plastic surgery, etc.)
4. Automate area security (locking/unlocking doors, alarms, etc.)
5. Provide a key tool for creating a more secure and controlled environment
6. Possible to provide target identification for military applications

Features

Our products features would include the following:

1. Accept hyperspectral image of a face as input
2. Lock/unlock doors wirelessly based on recognition
3. Ability to utilize GPU resources to achieve order of magnitude processing speed advantages over simple CPU based algorithms
4. Efficiently store a database of subjects
5. Efficiently access database for comparison purposes
6. Efficiently scale to different sizes of system processing resources
7. Output the top five matches upon completion

While our product has many benefits, we feel that it is important to note here that we are not going to be dealing with the sensor or automated preprocessing for input to our algorithm. We are simply handling the backend processing, data basing, and notification challenges.

II. Design

High-level Block Diagram

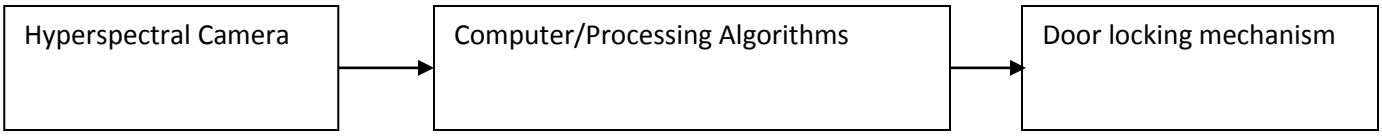


Figure 1. High-level Block Diagram

High-level Block Description

Hyperspectral Camera

This module is responsible for collecting the hyperspectral information to be processed. We will obtain preliminary data from a research group at Carnegie Mellon University[1] and a research group at Hong Kong Polytechnic University[2]. We also have a verbal commitment from Raytheon to get direct access to a hyperspectral camera in their facilities and obtain data from there. Although this module is an integral part of our design, we are not addressing its development in the scope of this project. We are only going to use data that has been collected with pre-existing systems. Therefore there is no lower-level description of this module.

Computer / Processing Algorithm

This module forms the bulk of our project. It utilizes a CPU, a GPU, and a database of hyperspectral information from multiple subjects. A description of the functionality of this module is given in the flow chart (Figure 2). The data contains an image of a person's face and is obtained from a hyperspectral sensor. It is in the form of varying intensities for different optical bands varying from 400-1100nm (visible to near-infrared region). We will be identifying a set of features whose hyperspectral information can uniquely identify a person. These features will be automatically extracted from the data. The algorithm will then process this information and compare it to an existing database. The algorithm will be developed using CUDA-C and parallelized using a GPU. An ordered list of top 5 matches will be produced for every test image. The door locking mechanism is engaged with the access level of the closest match. The access level is a number between 0 and 4 for every entity in the database.

The processing Algorithm is further described in the flowchart and detailed description below. The flowchart depicts how the algorithm will function. This is split into the feature ID algorithm, which explains the steps to select the best features and the flow of obtaining hyperspectral data from the pixels of these features, and the data comparison and processing algorithm that compares this obtained data to a database.

Door locking mechanism

This module will contain a wireless receiver and a microcontroller. These will interface with 4 motors through H-bridges. The microcontroller will control the H-bridges through its digital output pins. Each individual in the database will have a generic clearance rating associated between 0 and 4. When the algorithm is run, the best match and his or her clearance rating will be output to the microcontroller through the wireless receiver. The microcontroller will then output to the H-bridge circuits to spin 0, 1, 2, 3 or 4 motors, depending on the clearance rating. (For example, a clearance 0 match opens 0 doors, but a clearance 3 match opens doors 1, 2 and 3). The motors will only spin for a determined amount of time to open the doors and then stop. After 15 seconds, they will spin back in the opposite direction the same amount to close the doors. The door locking mechanism is further described in a block diagram (Figure 6) and description below.

Algorithm Flowchart

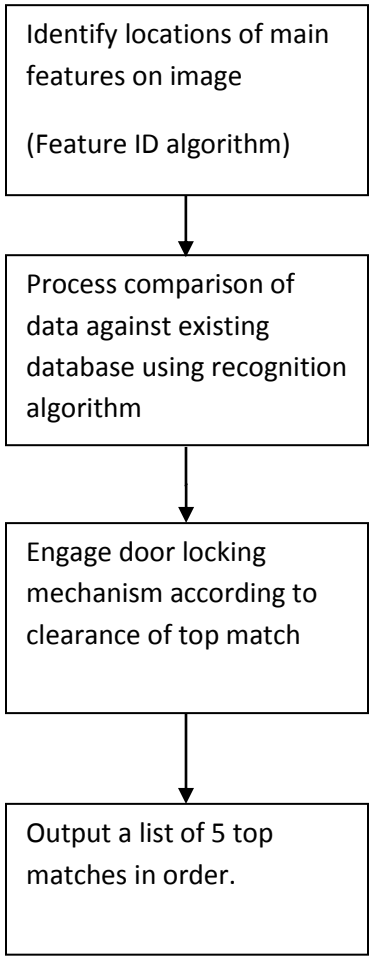


Figure 2. Algorithm Flowchart

Algorithm Flowchart Implementation

Feature ID Algorithm (Frist Block of Algorithm Flowchart)

The first part to development of the feature ID algorithm is to identify features, which can be used for unique identification. Typical computer vision technologies use a set of features from the eyes, forehead, nose, cheekbones, etc. This helps reduce the amount of data that needs to be stored for each subject and for our purposes, helps create a more parallelizable algorithm. The process behind selection of features would be to determine the ease with which the comparison features can be selected and if a combination of these features can uniquely identify a face. Past research papers[13] have shown that skin tissues like forehead, left cheek, right cheek, lips and hair are good choices for features to obtain unique hyperspectral data. However features like eyes and lips might be easier to extract and we can use them as geographic handles to extract these other features.

Once we determine which features are required, the next step is to figure out how to extract these features from the data. The algorithm will map the position of these features in the image. This will be done by geographically mapping the handler features like the eyes and then creating distance vectors based on the relative positions of the other features. We will select a square of N*N pixels around every comparison feature. This N will scale by the pixel distance between the handler features. The N*N square will be divided into M fixed number of blocks to use with the Mahalanobis algorithm as described in the data comparison and processing algorithm. For each of these M blocks, the spectral information from different bands of each pixel in the block, would be averaged, normalized and stored. This is the data that needs to be written to a database for each feature of every image: a set of M vectors, each vector containing average, normalized spectral information across a certain number of pixels, for the various bands.

Before performing the data processing and comparison algorithm, we will use this algorithm to populate our database with the required information to be stored for each subject. We will then take an image to be recognized, run this algorithm on it to extract the features, and then run the processing and comparison algorithm to look for matches in the database.

Feature ID Flowchart

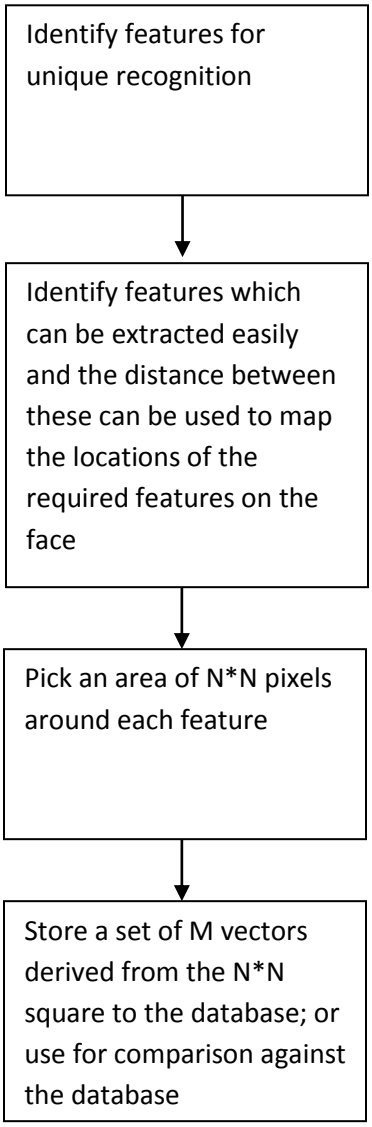


Figure 3. Feature ID Algorithm Break Down

Data Comparison and Processing Algorithm (Second Block of Algorithm Flowchart)

After taking the input data and processing it to extract sets of vectors describing relevant features, the data is passed to the Data Comparison and Processing algorithm, which is responsible for comparing the input image to the database and finding the closest match. The flowchart below (figure 4) describes the process of comparison used here. Essentially, we will be running a comparison of every database entry (a set of unique identifying features for a person) against our target data set in a feature-by-feature manner.

Looking at figure 4, we begin to notice already that this process lends itself nicely to parallelization since it contains nested loops, which do not have any conflicting read/write operations. Additionally, each operation preformed by these loops uses a constant and overlapping data set, which means that we can optimize device memory bandwidth to let us make the most use GPU processing power. It should be noted, however that we will likely opt to parallelize only the inner of these two loops and focus instead on the core comparison operation (a series of Mahalanobis distance calculations[12]) which lends itself even more optimally to parallelization (see figure 5 and preceding text).

Data Comparison and Processing Flowchart

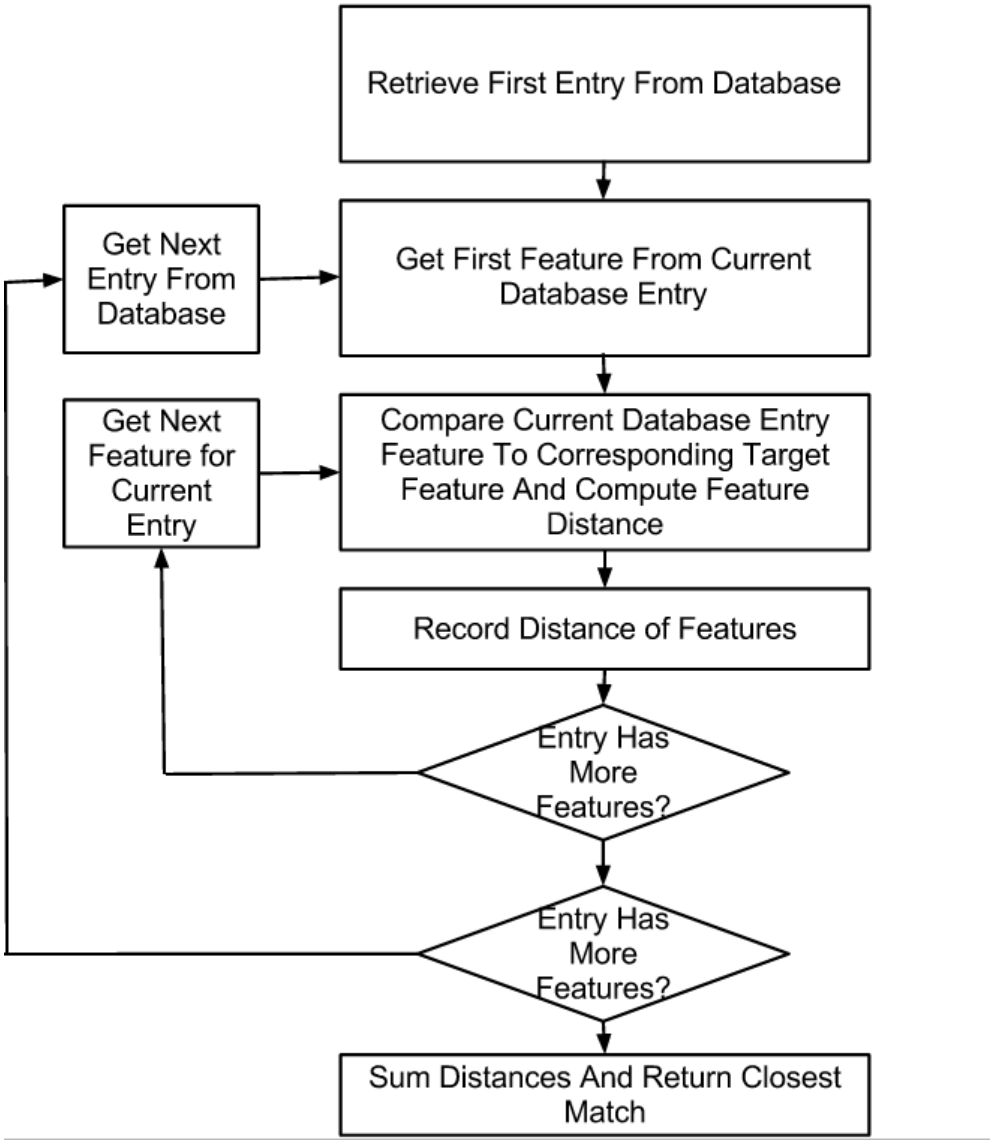


Figure 4. Data Comparison and Processing Breakdown

The Mahalanobis distance calculation[12] rests at the core of our data comparison and processing algorithm. This calculation assigns a distance between a set of vectors where a greater distance leads to a lower probability that the vectors are a reading of the same point. For each feature in our database entries and target, there is some number of normalized vectors M comprised of the average of groups of pixels read in by the feature ID algorithm (see previous section for description). We will use a Mahalanobis distance calculation to compare each vector from an individual feature on the target and its corresponding database entry feature (leading to M^2 calculations). From these calculations, we will find the two vectors with the shortest distance between the target and the entry and use this distance as our feature distance. This practice is to account for slight variations in selection of the feature area in the feature ID algorithm. We attempt to get somewhat consistent distance results between sampling a single subject by using this approach. Please see figure 5 for the calculation breakdown.

In examining figure 5, one will notice that once again, the algorithm presented here lends itself very nicely to parallel processing. The nested loop structure here will be fully parallelized, and since the calculations use each vector many times, loading them to the GPU device shared memory should allow us to optimize memory bandwidth usage in data transfer between the GPU and the host RAM. One challenge that will arise here will be storing the minimum of the computed distances. It is likely that we will simply create an array and export the task of selecting the lowest value from the array to the host processor, which runs quickly in single thread operation.

It should be noted here that our comparison algorithm is based on parallelizing the algorithm presented in the paper "Face Recognition in Hyperspectral Images"[13] and adding in functionality to select features by automation and machine vision rather than by hand. We will be optimizing implementation for speed and accuracy.

Once we have completed all of our distance calculations and found the minimum sum of Mahalanobis distances for the closest database entry to the target, we will make a list of the top 5 matches for every image. The correct image should be in the top 5 at least 50% of the time.

Testing and Training Databases

We will have at least 2 images of each person depending on the databases we are able to get access to. Depending on the size of our database we will use at least 5(total database size<30) to half (total database size>50) of total images for just testing and training purposes. The remaining images will not be tested or trained on until demo day. On demo day, we will load one set of the unused pictures into the database and perform the comparison with the other set.

Engaging The Door Lock (Algorithm Flowchart Following Second Block)

All entities in the database will be assigned a number from 0-4 depicting access levels. Depending on the access level of the closest match the doors will be opened by rotating

the required motors. Access level 0 will not open any doors, access level 1 will open door 1, access level 2 will open doors 1 and 2 and so on.

Feature Distance Calculation Flowchart

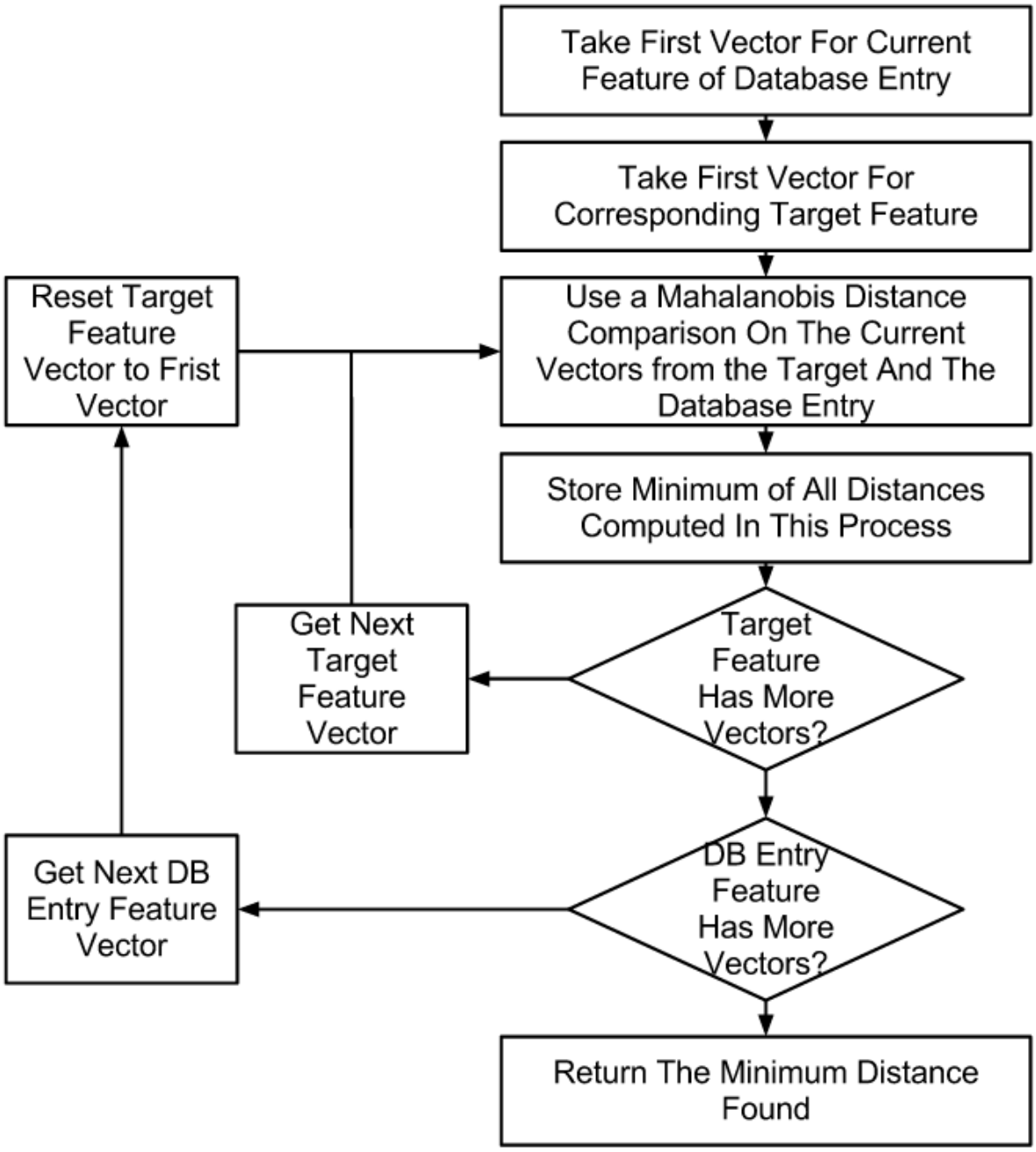


Figure 5. Feature Distance Calculation Breakdown

Door Locking Mechanism Block Diagram

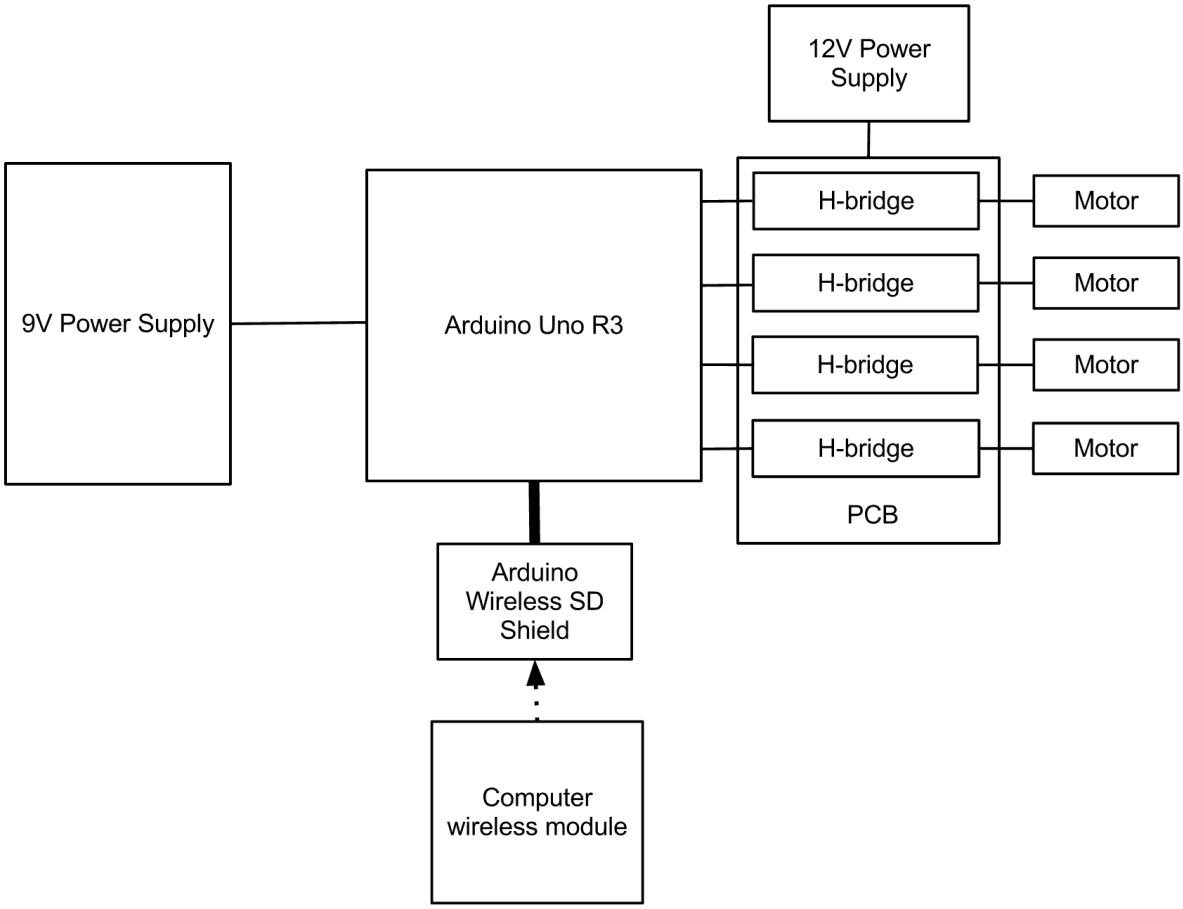


Figure 6. Door Locking Mechanism Block Diagram

Door Locking Mechanism Description

The door locking mechanism operates with these major components: A 9V power supply, an Arduino Uno R3, a wireless interface (Arduino Wireless SD Shield), the computer wireless module, a 12V power supply, a PCB containing 4 H-bridges, and four motors, each linked to one H-bridge on the PCB.

The 9V power supply powers the transistor circuit and Arduino. The Wireless SD module is powered through the Arduino (the Arduino plugs right into it). The motors and PCB are both powered by a 12V power supply, with the motors powered through the PCB's H-bridges.

The motors are controlled through the Arduino, which will output digital signals to the relevant H-bridges based on which motors need to be powered. For each motor, the Arduino can enable a forward and backward output. The H-bridges translate these two outputs into an operation on the motors to drive it forwards or backwards. Please refer to the H-bridge schematic (figure 9) in the

Schematics section for a description of the H-bridge functionality. The calculations needed for the selection of parts are listed in the Testing and Calculation section.

The Arduino we are using is an Arduino Uno R3. It is integrated with an Arduino Wireless SD Shield that connects to the computer through an Ad-Hoc wireless network to allow user and program input. The Arduino is programmed to do the following:

1) By default, keep the digital outputs 0-7 to low so the motors do not spin.

2) If the Wireless SD Shield receives an unlock signal for any motor, the Arduino should read the access level. Based on this access level, enable the forward operation digital output for required motors (0 access: no motors, 1 access: motor 1, 2 access: motors 1 and 2, etc.) Enable digital output 0 for motor 1, 2 for motor 2, 4 for motor 3, and 6 for motor 4.

3) After 3 seconds, the Arduino will disable any running motor. The locks will then be considered “unlocked”.

4) The Arduino will then wait for 15 seconds, and then spin the same motors backwards for precisely 3 seconds to lock the doors.

Door Locking Mechanism Wiring Diagram

The door locking mechanism has previously been explained. This is a wiring diagram with pin numbers for each chip and connections. The Arduino board does not have pin numbers; it has pin names as shown on the photo below (Figure 7).

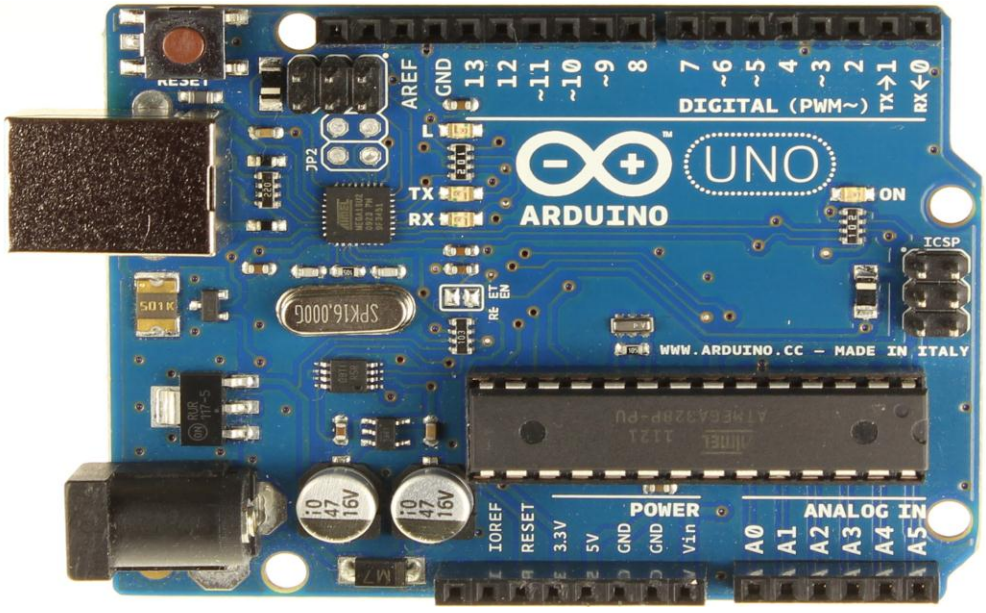


Figure 7. Photo of the front of Arduino Uno R3[3]

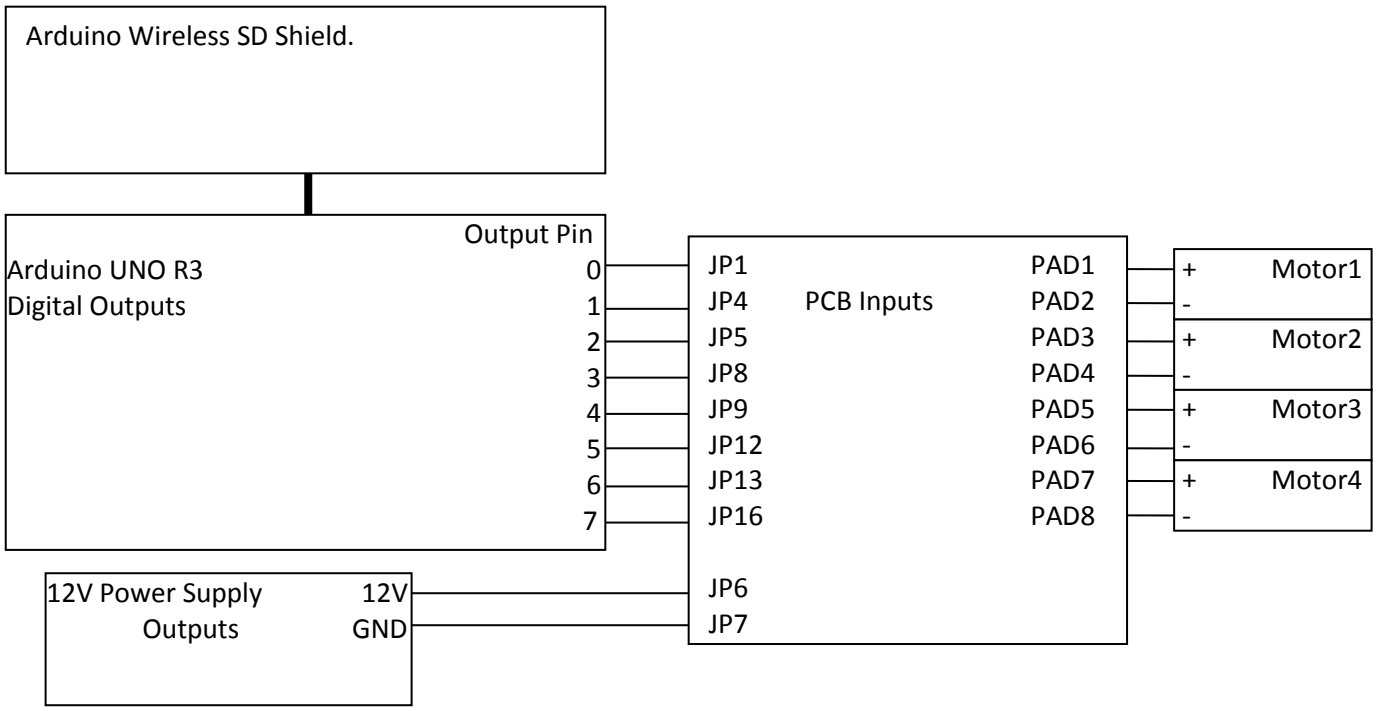


Figure 8. Wiring Diagram of the Door Locking Mechanism

The Digital outputs pins 0-7 of the Arduino Uno R3 go into the pins demonstrated by Figure 8 in the PCB. The PAD1 and PAD2 outputs from the PCB go into + and – of Motor1, PAD3, PAD4 outputs go into Motor 2, PAD5, PAD6 outputs into Motor 3 and PAD7,PAD8 outputs into Motor 4. The Arduino Uno R3 plugs straight into the wireless SD shield. The 12V power supply plugs into the PCB board to power the H bridge, through ports JP6 and JP7.

Testing and Calculations

Power Supply Calculations

The Arduino is powered by 6-20V DC (recommended 7-12V). This means that a 9V power supply is adequate to power our circuit.

PCB H-bridges Calculations

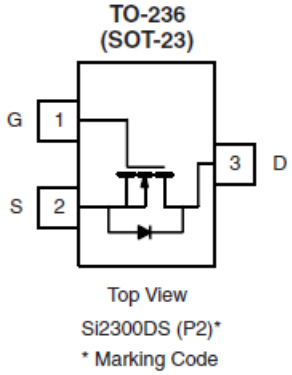
Calculations

For the Individual NMOS and PMOS FETs:

N Channel MOSFET – Si2300DS

Vt ~ 1.16volts at 25 degrees Celsius

Vg = 0 or 12volts (amplified signal from Arduino board)



Vs = Ground = 0volts

When Vgs > Vt NMOS is active
When Vgs < Vt the NMOS is in cutoff

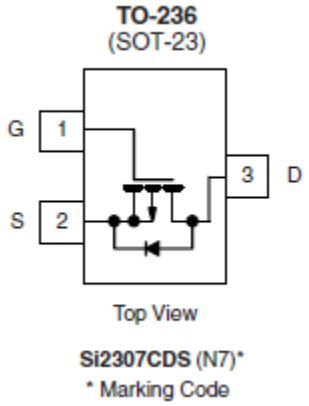
When Vg is high:
Vgs = Vg – Vs = 12volts
Vds ~ 0

When Vg is low:
Vgs = Vg – Vs = 0volts

Therefore the NMOS is in the cut off state when Vg is low and active when Vg is high.

P Channel MOSFET – Si2307CDS

Vt ~ -.1volts at 25 degrees Celsius
Vs = Vcc = 12volts (amplified signal from Arduino board)
Vg =0 or 12volts



When Vgs > Vt the PMOS is active.
When Vsg < Vt the PMOS is off.

When Vg is high:
Vsg = 12-12 = 0 volts

When Vg is low:
Vgs = 12 – 0 = 12 volts

Therefore the PMOS is in the cutoff region when Vg is high and in the active region when Vg is low.

Since our PCB contains four repetitions of the same circuit, we can analyze one H-bridge motor control unit and extrapolate the functionality of our entire circuit given that our power source is capable of outputting enough current.

For the single upper left H-bridge (from Figure 9):

When both Arudino inputs are low (JP1 low, JP4 low):

NMOS IC17 and NMOS IC23 will be in cutoff state causing a high voltage to be applied to PMOS IC1, PMOS IC3, NMOS IC2, and NMOS IC4. Both PMOS will now act as opens since their gate and source voltages are equal. The two NMOS not being directly driven by the Arduino will become active with the 12V gate

voltages and connect ground to both PAD 1 and PAD 2 (across the motor would be connected). This results in the motor not moving.

When the first Arduino input is high and the second is low (JP1 high, JP4 low):

NMOS IC17 will be in the active state resulting in a gate voltage of 0 for PMOS IC1 and NMOS IC2. NMOS IC23 will be in the cutoff state which will result in a 12v gate voltage for PMOS IC3 and NMOS IC4. This combination of gate voltages will result in PMOS IC1 and NMOS IC4 being active with NMOS IC2 and PMOS IC3 in cutoff. This will cause voltage to be applied across the motor and current flowing through the motor will cause it to spin. We have chosen to use a motor with a generous range of operating voltages (5 to 16) so falling inside this range is not a problem.

When the first Arduino input is low and the second is high (JP1 low, JP4 high):

This will be the same as for the last case, except the current will be applied in reverse and cause the motor to spin in the opposite direction. NMOS IC17, PMOS IC1, and NMOS IC4 will all be in cutoff. NMOS IC23, PMOS IC3, and NMOS IC2 will all be active.

When both Arduino inputs are high (JP1 high, JP4 high):

NMOS IC17 and NMOS IC23 will be active resulting in low input voltages for both PMOS and the two NMOS IC2 and NMOS IC4 being. The two PMOS will both be active while IC2 and IC4 will be in cutoff. This results in an even voltage on either side of the motor and no movement.

III. Schematics

PCB Board H-bridge circuitry

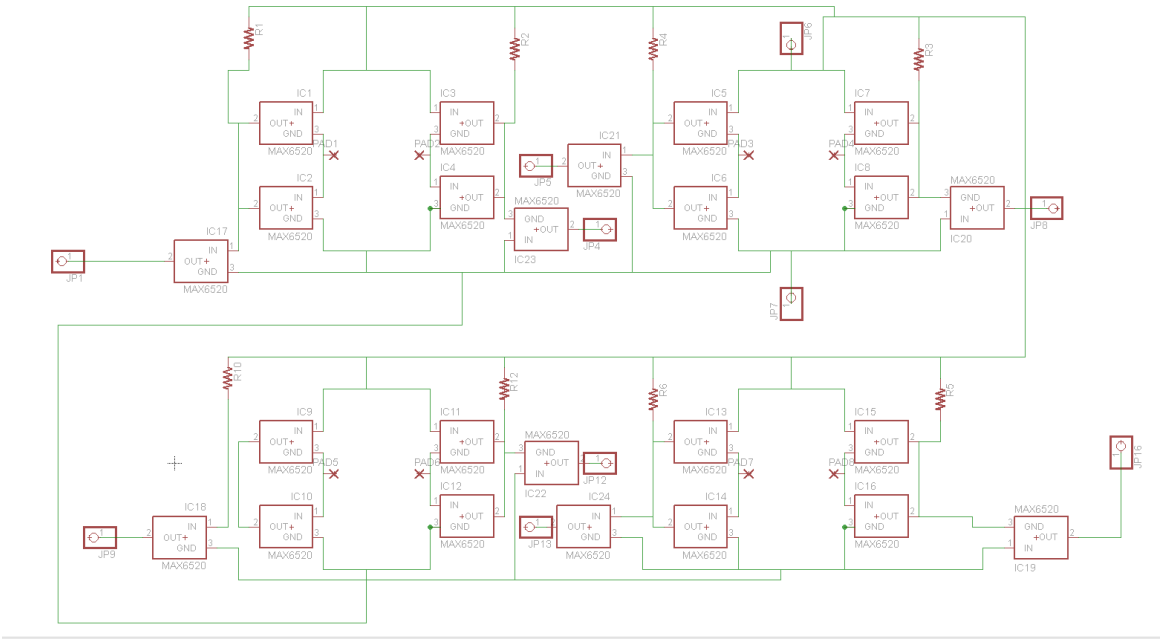


Figure 9. PCB Board Schematic

The PCB H-bridge circuit is set up to run the motors based on the inputs from the Arduino.

- JP6 is the VCC connection that comes from the 12V power supply. JP7 is the ground, also coming from the ground of the 12V power supply.
- JP1, JP5, JP9, and JP13 are connected to the forward-enable outputs from the Arduino pins 0, 2, 4 and 6 for motors 1, 2, 3, and 4 respectively.
- JP4, JP8, JP12, and JP16 are connected to the reverse-enable outputs from the Arduino digital output pins 1, 3, 5 and 7 for motors 1, 2, 3, and 4 respectively.
- PAD1 and PAD2 are the voltage + and – for motor 1 respectively, PAD 3 and PAD4 for motor 2, PAD5 and PAD6 for motor 3, PAD7 and PAD8 for motor 4.
- IC1, IC3, IC5, IC7, IC9, IC11, IC13 and IC15 are PMOSs with IN as source and GND as drain and IC2, IC4, IC6, IC8, IC10, IC12, IC14, IC16, IC17, IC18, IC19, IC20, IC21, IC22, IC23 and IC24 with IN as drain and GND as source.

The H-bridge consists of 2 PMOS and 2 NMOS transistors. The circuit also contains an NMOS transistor per input. This is because the inputs to the PMOS and NMOS on one side are pulled up to 12V when the input from the Arduino is 0V and they're ~0V when the input coming from the Arduino is 5V due to the high drop of voltage across the 10kohms resistors. Thus, when for example JP1 is high, the inputs to the PMOS and NMOS on the left side of the first H-bridge would be low, and vice versa.

In the H- bridge, the ones on the upper left(IC1-PMOS1) and the lower left(IC2-NMOS1) have the same input (input 1) while the ones on the lower right(IC4-NMOS2) and upper right (IC3-PMOS2) share a common input (input 2). When input 1 is high(12V) and input 2 is low(~0V), PMOS2 and NMOS1 are conducting and PAD1 output turns to low and PAD2 output turns to high. This makes the motor rotate in reverse. When input 2 is high(12V) and input 1 is low(~0V), NMOS2 and PMOS1 are conducting and PAD1 output turns to high and PAD2 output turns to low. This translates to the following truth table:

JP1	JP4	PAD1	PAD2	Motor operation
0	0	0	0	Stop
0	1	0	1	Reverse rotate
1	0	1	0	Forwards rotate
1	1	1	1	Stop

PCB layout

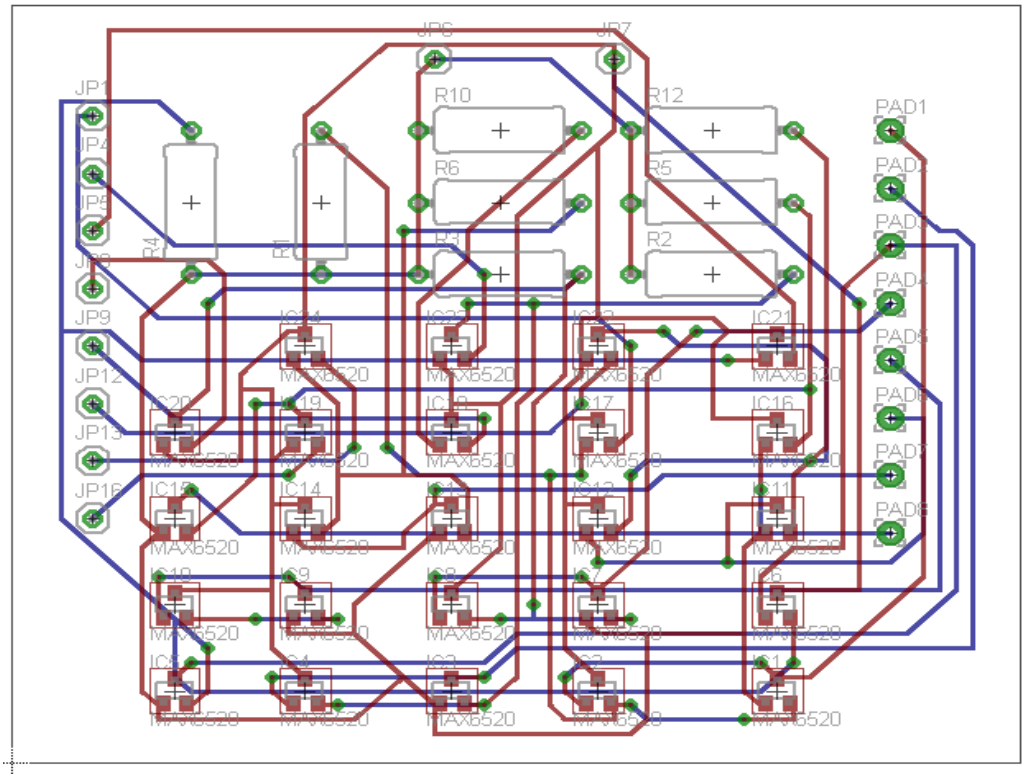


Figure 10. PCB Layout

This is the layout used to create the PCB board.

Arduino Uno R3 Schematic

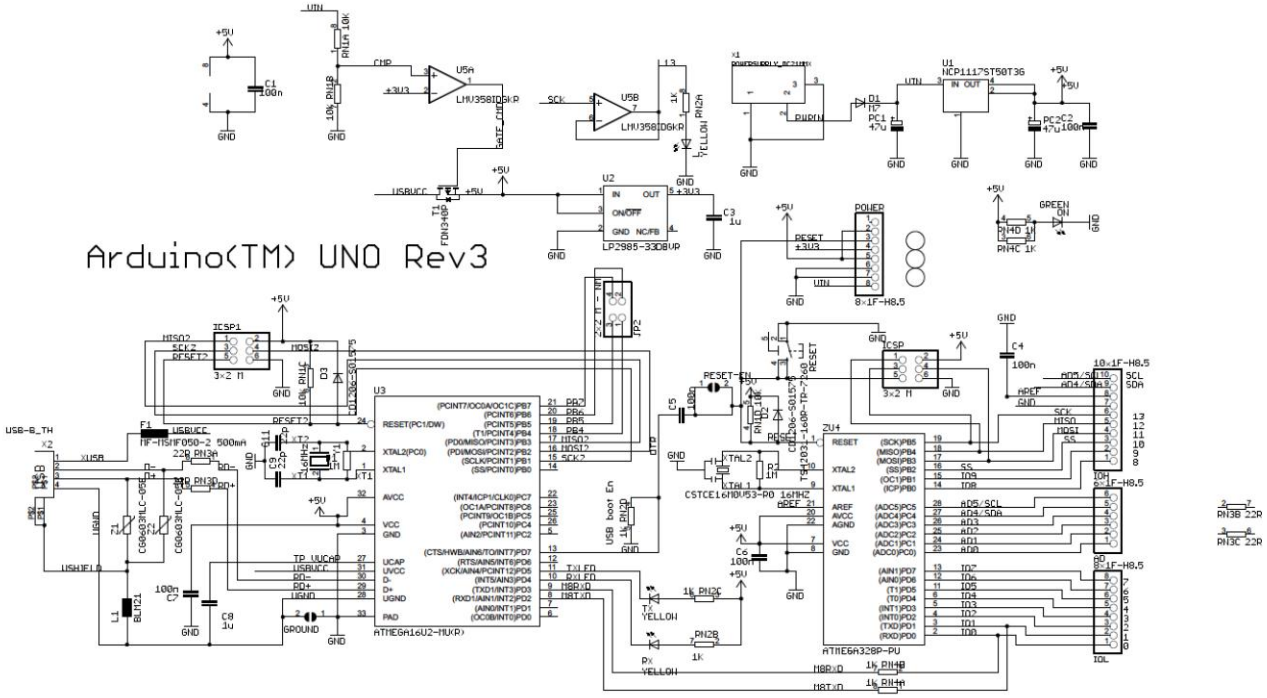


Figure 11. Arduino Uno R3 Schematic[5]

The Arduino schematic is used for reference purposes only to obtain pinning options. The Arduino Uno[6] is a microcontroller with "14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button". It operates at a voltage of 5V, and has an input Voltage range of 6-20V with a recommended input of 7-12V. The digital I/O pins output a current of 40mA and the 3.3V pins output a current of 50mA. There is 32KB of flash memory available, 2KB of SRAM, 1KB of EEPROM and the clock speed is 16 MHz. This available memory is more than sufficient for the purposes of our program which we estimate will require under 150 lines of code, thus using only a fraction of the approximate 300,000 that the memory on the Arduino provides.

Arduino Wireless SD Shield Schematic

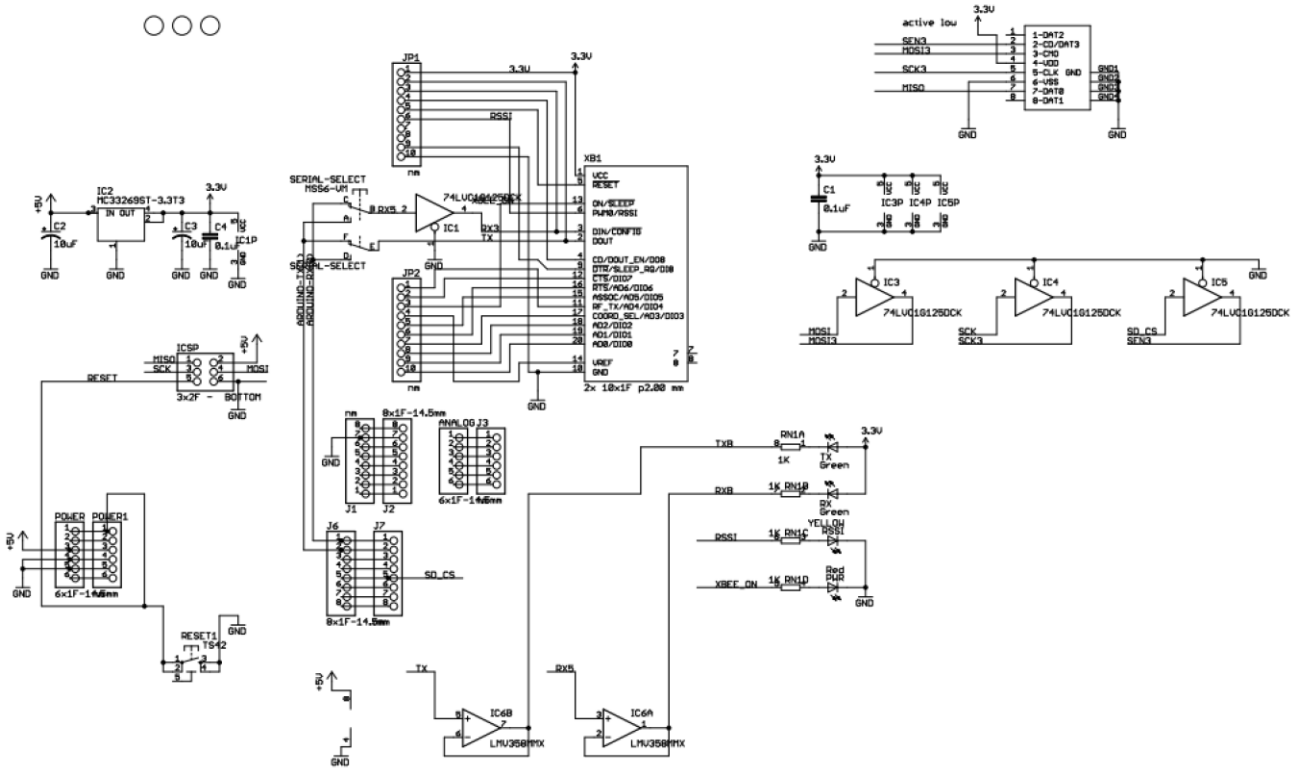


Figure 12. Arduino Wireless SD Shield Schematic[7]

The Arduino Wireless SD shield[8] allows an Arduino board to plug right into it and interact wirelessly with any wireless device. The range on the wireless is up to 100 feet indoors and up to 300 feet outdoors (with line of sight). We plan on limiting our device to a functional 25-30 feet, so this range is more than adequate for that purpose. The Wireless Shield is used as a replacement for the serial/USB input of the Arduino board in our design.

IV. Requirements and Verification

High-Level Requirements, Verifications, and Testing Procedures

For the sake of academic integrity, all verifications must be run and passed with the demo database and not the training database used for our own tests.

Block	Requirement	Verification
Hyperspectral camera data	Data must be hyperspectral - Wavelength of data between 400 and 1200 nanometers. - Intensity information is present in different bands.	- **Testing Procedure: Plot a sample of the data with reflectance as a function of wavelength. Ensure that the data has intensity information present across a range of at least 300nm of wavelengths between 400 and 1200nm.**
Hyperspectral camera data	Data can be recognized by the computer - Data is in a file format readable by the Linux operating system - Data can be read and used by a CUDA program	- **Testing procedure: Write a simple CUDA program that opens a file from the data's format, reads the content of that file and prints these contents in any form to the screen.**
Computer / Algorithm	Databases are sufficiently and correctly populated. - **Each database (testing and demo) has at least 20 unique entries.** - **Each entry has hyperspectral intensity information for each feature.** - **Each Database includes at least 50 total entries including automated generated data.**	- **Testing Procedure: Open each database on the computer. For each database:** - **Use the computer's database software to output the total number of elements and ensure it is greater than 50.** - **Also output the number of unique elements and ensure it is greater than 20.** - **Finally, run a database query to output the number of elements with incomplete fields. Ensure that this returns 0.**
Computer / Algorithm	Features are extracted correctly	- **Testing procedure: run the following test 5 times: (need 4 positive results)** - **Input a hyperpectral image to the algorithm. Pre-identify locations of comparison and handler features to be extracted by the algorithm. Use print**

		statements in the code to output the algorithm's identified location of these features. Verify that handler and comparison features identified by the algorithm are within a 10% of image dimension range from the pre-identified locations.
Computer / Algorithm	Comparison of a new picture of an individual with an old, existing picture in the database of the same individual under similar lighting/orientation conditions should ideally yield that individual as best match	- **Testing Procedure: Run the following test 10 times:** - **Obtain two hyperspectral images of an individual with the same facial orientation and lighting. Place one in the database, with at least 19 other unique photos also in the database. Run the second test image with the program. Make sure the right photo is selected in the top 5 at least 50% of the times.**
Computer / Algorithm	The door locking mechanism gets the required information from the wireless packet for the access level of the top detected match.	- **Testing Procedure: Pass through 20 photos to test.** - **Run the algorithm with an image that exists in the database. Use a print statement in the code to output the user's access level to the screen. Make sure the correct motors have rotated based on the access level.**
Computer / Algorithm	The algorithm should output its top 5 matches in order and the subject numbers of the people associated with them. The target should be in those top 5 matches 50% of the time.	- **Testing Procedure: Output the subject number of the target. Output the subject numbers of the people associated with the top 5. Use an indicator on the screen to indicate if the target is in the top 5 or not.**
Door Locking Mechanism	The door locking mechanism must properly receive positive signals from the computer block.	- **Testing Procedure: Write the Arduino with a simple program that outputs a 5V voltage to pin 1 when a signal is received. Send the signal through the computer, use a voltmeter at pin 1 to verify the voltage switches to 5V.**

Door Locking Mechanism	If a positive signal is properly received from the Computer Block, the door locking mechanism must engage the motors	- **Testing procedure: Pass a signal corresponding to each access level to the door locking mechanism; ensure that the proper motors rotates any amount.**
Door Locking Mechanism	The motors must rotate the correct amount	**- Testing Procedure: Pass an image that exists in the database with security level 4. Ensure that all four motors rotate for 3 seconds in one direction, stop, wait for 15 seconds, and rotate for 3 seconds in the other direction.**

Hyperspectral Camera / Data

Due to the nature of this block, no further breakdown in verification is required. If all high-level requirements are met by the data we use, it will ensure proper functioning of this block in the overall design. No lower-level requirements can be identified.

Computer / Algorithm

The algorithm must perform as declared by the high-level requirements. An additional breakdown of requirements is possible using the flowchart to step through the algorithm by functional component. To avoid redundancy, refer back to the blocks in figures 2 through 5. The test procedure will simply be passing in a test input with a known output and verifying proper output for each code block.

Door Locking Mechanism

The door-locking mechanism is composed of two Power Supplies, A PCB board with 4 H-bridges, the Arduino Uno R3 microcontroller, the Arduino Wireless SD Shield, and four electric motors. Below are detailed the requirements for all of these Components. All voltage checks must be true with a 20% lenience each way.

Block	**Requirement**	**Verification**
Arduino Uno R3	The Arduino Uno R3 microcontroller must turn on, turn off, and be reprogrammable	- **Testing procedure: Turn on the Arduino, make sure the power LED turns on, turn off the Arduino.** - **Program the Arduino to output 5V in pin 5, and output 2V in pin**

		6. Check voltages with multimeter.
Arduino Uno R3	The Arduino Uno R3 must consistently output a high signal and low signal through the I/O pin when desired.	- **Testing procedure: program the Arduino to output 5V for 10 seconds at pin 5, and 0V for 10 seconds alternatively (square wave). Ensure with a oscilloscope at pin 5 that the correct wave is produced.**
Arduino Wireless SD Shield	The wireless SD shield must connect to the computer and be able to receive signals from it and pass these signals through to the Arduino.	- **Testing procedure: write a simple program that sends a signal to the board. Program the Arduino to output 5V at pin5 when that signal is received, 0V if it is not. Probe pin5 and run the program, make the reading switches between 0V and 5V.**
Motors	The motor should rotate in the correct direction when the inputs are applied to it.	- **Testing Procedure: Apply a high voltage (9V) at Pad1 and 0V at Pad2 using multimeter. The motor connected should rotate in the 'forward' direction. On switching the inputs to Pad1and Pad2, it should rotate in the reverse direction**
PCB H-bridges	The H bridge must output the correct voltages to the motor it is connected to according to the 2 input signals	- **Testing Procedure: Apply a high voltage (5V) at JP1 and 0V at JP4. The output at Pad1 should be high and at Pad2 should be low. On switching the inputs to JP1 and JP4, the outputs should now reverse.**
9V Power Supply	The power supply must provide adequate power of 9V to the Arduino.	- **Testing Procedure: Set the power supply to output 9V. Ensure that over a period of 2 minutes the power supply does output 9V with up to 10% fluctuation using a voltmeter.**

12V Power Supply	The power supply must provide adequate power of 12V to the PCB.	- **Testing Procedure: Set the power supply to output 12V. Ensure that over a period of 2 minutes the power supply does output 12V with up to 10% fluctuation using a voltmeter.**

PCB circuit

Block	**Requirement**	**Verification**
PMOS	The transistor should operate correctly when the right biases are applied.	- **Testing Procedure: Apply JP1=0V. Hook up an LED from PAD1 to ground via a 500ohm resistor. If the LED lights up when JP1=0V, and turns off when JP1=5V, the PMOS transistor is working fine.**
PMOS and NMOS circuit	The transistors should operate correctly when the right biases are applied.	- **Testing Procedure: Apply JP1=5V and JP4=0V. Hook up a bidirectional LED between PAD1 and PAD2 with a 500ohm resistor. If the LED lights up red, and green when JP1=0V and JP4=5V, the network is working fine. Apply JP1=5V and JP4=5V, and JP1=0V and JP4=5V. The LED should not light up in both cases.**
Resistor	The resistor must be within the required range from its desired resistance	- **Testing Procedure: Use an Ammeter to test the resistance of the resistor. Ensure that it is within 10kohm+-10%.**
Connectivity Testing	All wires on the PCB should be connected properly.	- **Testing Procedure: Do a resistance probe with an ohm meter. Make sure the resistance is less than 20 Ohms between things that are supposed to be connected, and above 1 MOhm between things not supposed to be connected according to Figure 9.**

Tolerance Analysis

Computer / Algorithm

We will run our algorithm with the data we have from actual hyperspectral images as well as automatically generated data for speed testing purposes (it is not possible for us to get a very large data set on the order of hundreds or thousands of subjects). Two major components affect the performance of our system the most, both related to the algorithm.

The first component is accuracy. The testing procedure for this would be to run hyperspectral images against the database. The requirement is a minimum of 20 subjects, each with an existing image in the database and a different image for testing, and a minimum of 180 other database entries (automatically generated). In these conditions, the algorithm should correctly have the subject in the top 5 matches at least 50% of the time.

The second component is speed. Because we are developing a basic parallel algorithm, the speed of the algorithm using a GPU should be greater than when using the CPU. Thus, we will run the algorithm on the computer without the utilization of the GPU and compare it to the speed with a GPU. To obtain a good benchmark, we will use automatically generated data in addition to our obtained data to produce a large enough data set.

Door Locking Mechanism

The voltages and currents must be fairly constant throughout in the door locking mechanism. We can assign a 20% tolerance to the 5V output of the Arduino, a 20% tolerance to the power supply voltage since a wide range of voltages can power the Arduino, a 20% tolerance in the voltage output from the power supply to the H-bridge and a 10% tolerance in the resistance of the resistors. Our selected motor can operate from 5-16V. The 12V power supply should be capable of sinking at least 1A of current to provide ample power to turn on and power our motors. Each motor has a starting current of 198mA of current according to the specification sheet and draws a minimum of 15mA.

V. Cost and Schedule

Cost Estimates:

We expect our main costs aside from labor to come from additional data collection in this project and building a computer dedicated for testing our algorithm. We expect our total project cost not including labor to be significant, but we will be receiving sponsorship from Raytheon Space and Airborne Systems. Our current plan is to travel to California and use one of Raytheon's labs and sensor setups to collect our data.

Labor:

Name	Rate/Hour	Overhead(x2.5)	Hours*	Total
Chris	$45	$112.5	240	$27,000
Timothee	$45	$112.5	240	$27,000
Akshay	$45	$112.5	240	$27,000
			Total Labor Cost:	$81,000

*Assuming a 12 hr. work week for 12 weeks

Parts:

** = Financial support expected from Raytheon SAS via the university*

*** = Support expected in kind directly from Raytheon SAS*

Door Locking Mechanism

Part/Service	Quantity	Cost Estimate	Potential Supplier	Total
Arduino Uno R3 Microcontroller* (1)	1	$23.76	Amazon	$23.76
Arduino Wireless SD Shield (2) *	1	$25.95	Amazon	$25.95
NMOS Transistors (3)	20	$0.5	Digikey	$10
PMOS Transistors (4)	10	$.6	Digikey	$6
500ohm resistors	10	$.5	ECE Parts shop	$5
Bidirectional LEDs	5	$.50	ECE Parts Shop	$2.50
10kohm resistors(5)	25	$.334	Digikey	$8.35
Motors (5)	5	$6.44	Digikey	$32.20
			TOTAL:	$113.76

(1) http://www.amazon.com/Arduino-Rev-3-Uno-R3/dp/B006H06TVG/ref=pd_bxgy_pc_text_y

(2) http://www.amazon.com/Arduino-Wireless-SD-Shield/dp/B006RATC2E/ref=sr_1_1?ie=UTF8&qid=1349059529&sr=8-1&keywords=arduino+wifi+shield

(3) http://www.vishay.com/docs/65701/si2300ds.pdf

(4) http://www.vishay.com/docs/68768/si2307cd.pdf

(5) http://www.digikey.com/product-detail/en/1622796-6/A105970CT-ND/3477555

(6) http://www.nmbtc.com/pdf/motors/MXN12.pdf

Computer Platform for Running Algorithm

Part/Service	Quantity	Cost Estimate	Potential Supplier	Total
Mother Board*	1	$110	Newegg.com	$110
8G Ram*	1	$50	Newegg.com	$50
i5 CPU*	1	$250	Newegg.com	$250
Nvidia GPU*	1	$1100	Newegg.com	$1000
Case*	1	$100	Newegg.com	$100
Hard Drive*	1	$200	Newegg.com	$200
			TOTAL:	$1710

Additional Data Collection (The current plan is for the team to travel to California where Raytheon has sensors in place that we can use cheaply rather than deal with shipping costs and risks etc. We will attempt to estimate the value of contributions here.)

Part/Service	Quantity	Cost Estimate	Potential Supplier	Total
Lab Usage**	1	$2000	Raytheon SAS	$2000
Technician labor time**	1	$1000	Raytheon SAS	$1000
Plane travel to and from site *	3	$500	Raytheon SAS	$1500
Hotel*	3	$150	Raytheon SAS	$450
Food during travel*	1	$600	Raytheon SAS	$600
			TOTAL:	$5550

TOTAL PARTS COST = $5550+$113.76+$1710 = $7373.76

TOTAL COST = $81000 + $7373.76 = $88373.76

Raytheon will directly fund the computer hardware, door locking mechanism hardware and costs of trip for data collection. This amounts to a total direct funding of $4646. Raytheon will indirectly fund the costs of lab and technician during the data collection trip.

In case of unforeseen circumstances, our total necessary parts cost is only for the door locking mechanism. The processing requirements can be met by utilizing personal and university computing resources. We can use data from the Hong Kong Polytechnic University and Carnegie Mellon University to compensate for collection of data, and we will still be able to have a functional product.

Schedule

Week	Chris Baker	Tim Bouhour	Akshay Malik
9/16	Work on proposal.		
9/23	Design Review: Recognition algorithm design lead. Aid with lock mechanism, feature ID algorithm, and system level development. Obtain data sets.	Design Review: Lock mechanism design lead. Aid with recognition algorithm, feature algorithm, and system level development.	Design Review: Feature ID algorithm design lead. Aid with recognition algorithm, system level design, and lock mechanism development.
9/30	Edit Design Review. Follow up on obtaining data sets.	Edit Design Review.	Edit Design Review
10/7	Begin Purchasing Parts. By Friday make decision based on status of Raytheon funding and purchasing. Research best features for recognition. Begin work on parallel recognition algorithm implementation focusing on feature reference points. Finish acquiring data type.	Begin Purchasing Parts. By Friday make decision based on status of Raytheon funding and purchasing. Research into processing data into a uniform format. Research best features for recognition. Begin Research into feature ID algorithm.	Begin Purchasing Parts. By Friday make decision based on status of Raytheon funding and purchasing. Research wireless communication between the lock and computer. Begin research into feature ID algorithm.
10/14	Finish acquiring all parts. Work on parallel recognition algorithm development. Research into accounting for data source variation based	Finish acquiring all parts. If needed, assemble computer. Install Linux operating system and programs needed for development	Finish acquiring all parts. Construct lock assembly. Set the lock successfully from computer. Work on feature ID algorithm

	on light and other factors.	in C++, CUDA C, and MATLAB. Help Construct lock assembly.	implementation.
10/21	Individual Project Report. Continue coding of parallel recognition algorithm. Support debugging of feature ID algorithm.	Individual Project Report. Set up database with hyperspectral data. Work on feature ID algorithm implementation. Research into accounting for data source variation based on light and other factors.	Individual Project Report. Help set up database.Work on feature ID algorithm implementation. Research into accounting for data source variation based on light and other factors.
10/28	Finish Debugging feature ID recognition algorithm. Work on parallel recognition algorithm implementation.		
11/4	Finish parallel recognition algorithm implementation.	Finish parallel recognition algorithm implementation. Focus on integration of database.	Finish parallel recognition algorithm implementation. Focus on integration of feature ID algorithm.
11/11	Write serial version on parallel recognition algorithm and run speed comparison test against parallel algorithm.		
11/18	Work on presentation/ compare performance further between serial and parallel/ make optimization tweaks.		
11/25	Tentative on site data collection in Raytheon SAS labs/ format data to be accepted by feature ID algorithm /testing and debugging of algorithms		
12/2	Further systems integration and test including lock hardware; debugging. Final Systems tweaks/ demo/ presentation/ final report.		
12/9	Schedule Buffer/ finish report.		

VI. Ethical Considerations

IEEE Code of Ethics

The first pledge of the IEEE code of ethics is to "to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;" Our project follows this pledge by creating a technological solution that can increase public safety in a variety of applications (crime fighting, security checkpoints, counter-terrorism).

The third pledge of the IEEE code of ethics is to "to be honest and realistic in stating claims or estimates based on available data;" Our project embodies this by clearly stating the limits and range of our end result, as well as clearly defining the scope of available data and making our claims accordingly.

The ninth pledge of the IEEE code of ethics states "to avoid injuring others, their property, reputation, or employment by false or malicious action;" In this project, we recognize the potential for harm due to false alarms. Therefore, we have set up a clear percentage of accuracy to nuance the credibility of our efforts and clearly identify the limits and imperfection of our tool. This will help avoid any incorrect matches leading to endangering others or their reputations.

Academic Honesty

We pledge to uphold the strictest standards of academic honesty; to not take credit for the work of any others, clearly state sources, and give credit where credit is due.

To maintain integrity of results, we will have at least 2 images of each person depending on the databases we are able to get access to. We will have separate databases for training and demoing. The demo images will not be tested or trained on until demo day. On demo day, we will load one set of the unused pictures into the database and perform the comparison with the other set. No training images will be used for verification or demoing.

VII. Citations

Sources

[1] Louis J. Denes, Peter Metes, and Yanxi Liu, "Hyperspectral Face Database," tech. report CMU-RI-TR-02-25, Robotics Institute, Carnegie Mellon University, October, 2002

[2] Wei Di, Lei Zhang, David Zhang, and Quan Pan "Studies on Hyperspectral Face Recognition in Visible Spectrum with Feature Band Selection" *IEEE Trans. on System, Man and Cybernetics, Part A*, vol. 40, issue 6, pp. 1354 – 1361, Nov. 2010

[3] Photo of the front of the Arduino Uno R3 http://arduino.cc/en/uploads/Main/ArduinoUno_R3_Front.jpg

[4] Bashford, Anthony J. Patent: Electric Strike Assembly. 04/05/2005. http://www.freepatentsonline.com/6874830.html

[5] Image: Arduino Uno R3 schematic. http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf

[6] Arduino Uno R3 official reference page. http://arduino.cc/en/Main/ArduinoBoardUno

[7] Arduino Wireless SD Shield schematic. http://arduino.cc/en/uploads/Main/arduino_WirelessShield_SD_v3-schematic.pdf

[8] Arduino Wireless SD Shield official reference page. http://arduino.cc/en/Main/ArduinoWirelessShield

[9] Open CV for image processing: http://opencv.willowgarage.com/wiki/

[10] H bridge: http://en.wikipedia.org/wiki/H_bridge

[11] Sedra, Adel S., and Kenneth Carless. Smith. *Microelectronic Circuits*. New York: Oxford UP, 1998. Print.

[12] Description of the Mahalanobis Distance Calculation: http://en.wikipedia.org/wiki/Mahalanobis_distance

[13] Zhihong Pan; Healey, G.; Prasad, M.; Tromberg, B.; , "Face recognition in hyperspectral images," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.25, no.12, pp. 1552- 1560, Dec. 2003