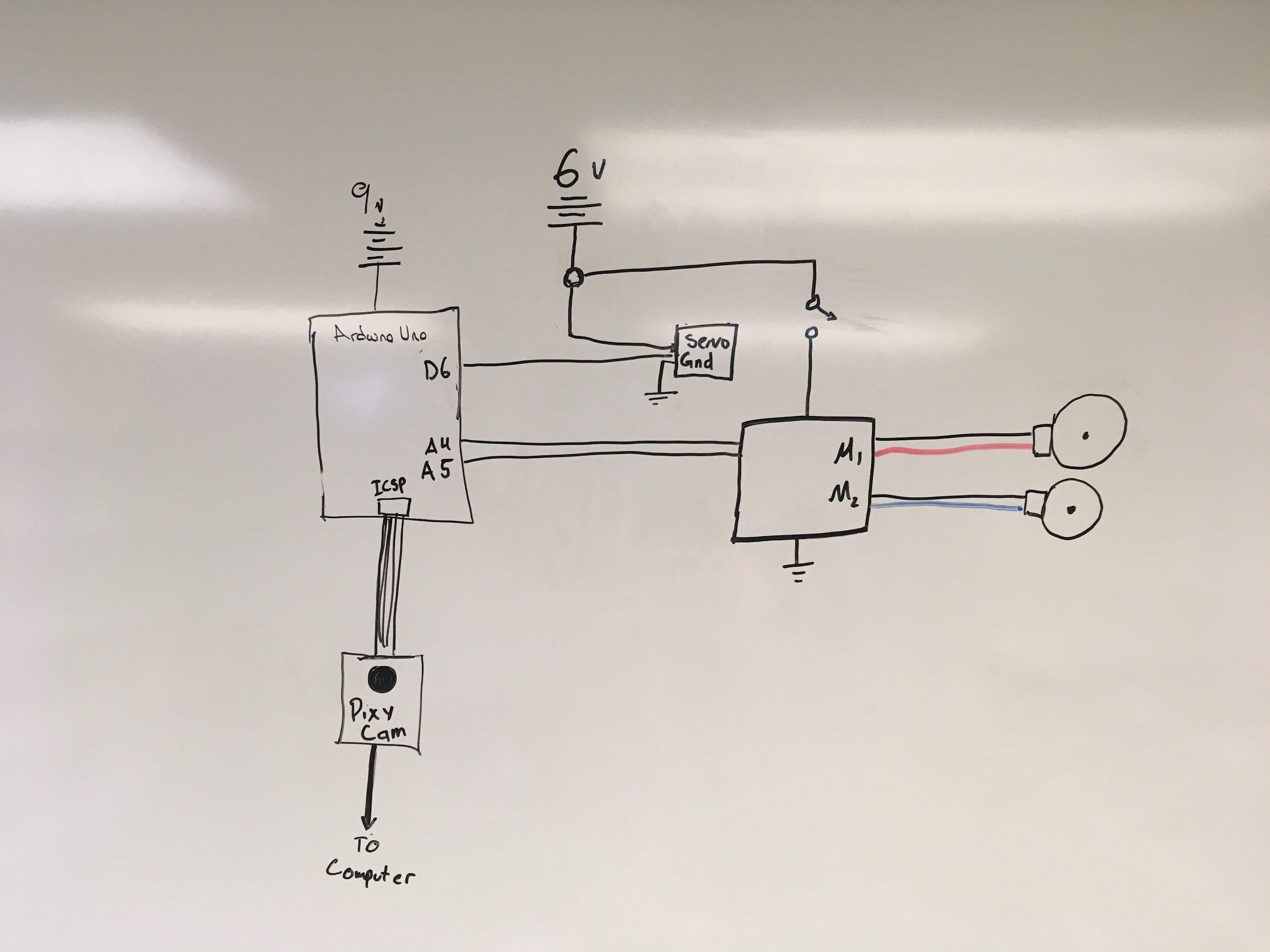
The Warehouse Bot created by Hugh, Ben, and Jehan is an electronics project using an Arduino, Driving Robot, and Pixycam. This is achieved using the Pixycam to track the coordinates of two bases and the bot using color tracking. It then moves the driving robot to the base based on the angle of the robot and the position of the base. The robot picks up the ball using a servo claw and moves to the final base destination. Once completed, the claw releases the ball and moves away from the base.

**Video link:**

[**https://www.youtube.com/watch?v=bzh7q3fI6iQ&feature=youtu.be**](https://www.youtube.com/watch?v=bzh7q3fI6iQ&feature=youtu.be)

**Circuit Schematic:**

****

**Detailed explanation:**

The warehouse bot uses the existing software and hardware that was used in the turtle robots in the lab 9, where both a follower and chicken programs were developed. For the warehouse bot, the goal of the project was to mimic the automated process methodologies used in warehouses where a given payload is located at a shipping base and carried to the destination base where it is dropped off. To implement this into an Arduino environment, our group used a pixy cam to track the needed information of two bases and the bot using color-coded pieces of paper that can be located anywhere within the camera’s field of view. The needed information from the pixy cam is the x and y coordinates of both bases, along with the position and angle of the bot.

Initially, our group tried to implement a completely untethered product using Bluetooth chips by using two Arduinos by having one on the robot running the program, and another controlling the pixy cam that would send data wirelessly. This would be the ideal situation, though implementing Bluetooth has a number of disadvantages (related to latency, accuracy, range, documentation, availability, and resources) that made it impossible to complete given the amount of time and resources available. All in all, using Bluetooth with the warehouse bot was not feasible, so the only other option available was a tethered version of the final product. We have included the code as well. This code would do all of the calculations for the bot as shown below and then send the information as four bytes of data which would then tell the bot where to turn.

From a hardware perspective, the circuit constructed consisted of an Arduino Uno connected to the pixy cam via the ICSP ports that were powered from a computer source. Analog pins A4 and A5 were connected to a pair of DC motors to the Arduino, supplied by a 9V battery. These motors are also connected to a switch that powers the wheels on and off for ease of use. Lastly, the servo claw is connected to pin D6 that is powered by the same 9V supply. The code used in this project was heavily based off of the Turtle Bot lab, where our group was able to recycle the non-menu functions and libraries previously used. Briefly going through the code, we start our program by defining what libraries and variables used on the top. This is followed by implementing servo object, serial display, and motor shield initialization in the void setup followed by void loop. In void loop, we define a while loop for the bot going to the ball, and command our Arduino to collect the info on the object tracked by the Pixy. The serial printing part of the program is also written that prints the collected info to the serial monitor. It then checks that if the number of objects is greater or equal to two, collect the given x and y coordinates of the bot and ball. If the correct number of objects are not being tracked, the program stops. The program then finds the distance between the ball and bot relative to the x and y positions, and finds the distance between them and calculates a speed proportional to the displacement of the angles (that can be positive or negative). It then calculates that if the speeds are within a certain range, turn right or left and when the angle is within a certain range execute the next function (Bot turn to ball, bot drive to ball, then same for bot to base). For driving to the ball/base, the bot’s wheel speeds are proportional to the maximum of the difference between the x-coordinates and y-coordinates of bot and the ball/base. Along with this, it is still calculating through the angle difference so that the bot can correct its trajectory as it goes. The claw also closes within a certain range of the desired object. Our group then used the same code that drives the bot from the initial base to the destination with a few slight changes that primarily opens the claw and backs up the robot and executes again for another run or object.