Color Programmable Control Board

Team 63
Anthony Shvets and Zhe Tang

Introduction

Opportunities for young kids to learn engineering and programming skills are ever expanding although there are some gaps to be filled.

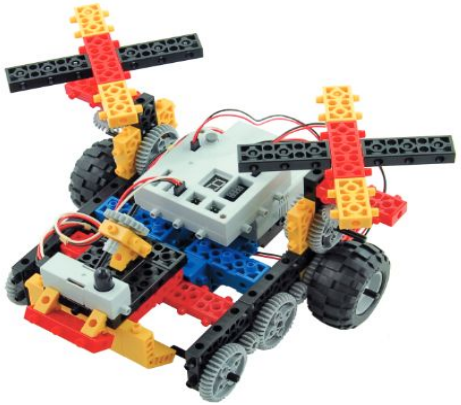
Kids can program with static robotics models in color and easily build robots with platforms like LEGO, although not both.

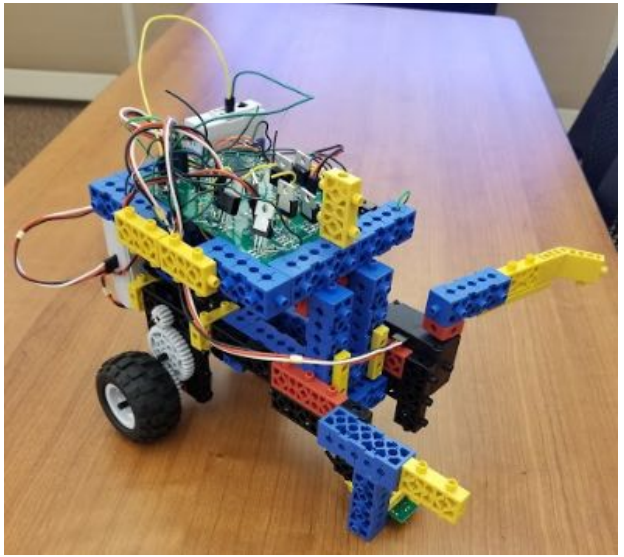
Objective

To create a control board and sensor

Would interface with a set of two DC motors and a servo motor

Useable with building platform that is flexible enough to accomodate varieties of different build structures





Objective

General Colors:

Background = White

Standard Line = Black

Color Code Words:

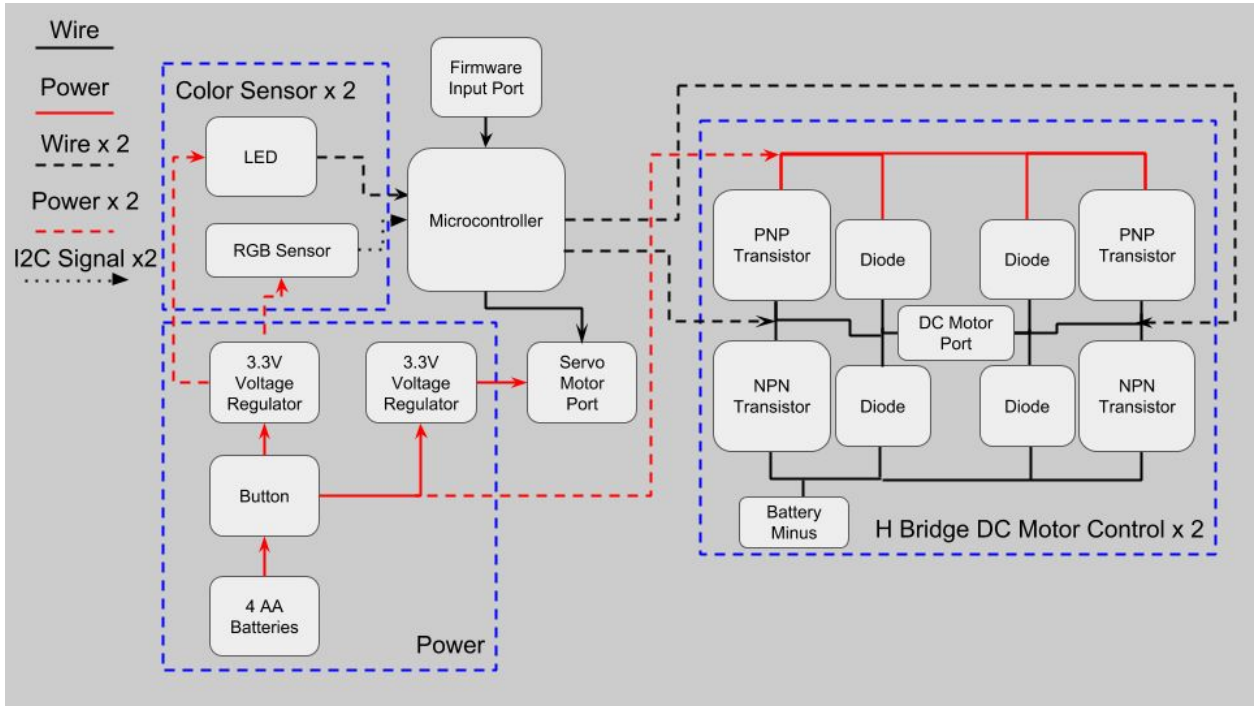
Stop Moving =	Blue	Red	Green	Red
Turn Around =	Blue	Red	Green	Brown
Servo 0 =	Blue	Green	Red	Green
Servo 45 =	Blue	Green	Red	Brown
Servo 90 =	Blue	Green	Brown	Red
Servo 135 =	Blue	Brown	Green	Brown
Servo 180 =	Blue	Brown	Green	Red

No words can have back to back colors of the same type

Original Design

Major Elements:

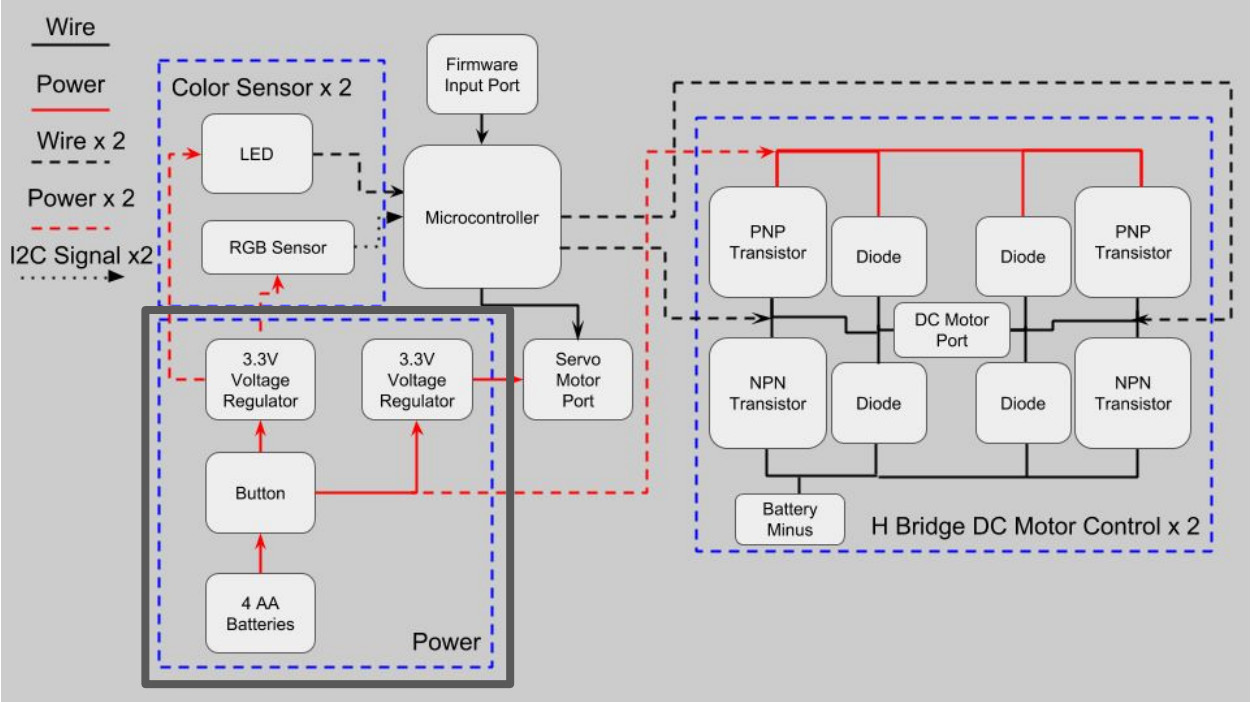
- Power Module
- DC Motor Control
- Color Sensors
- Microcontroller (MCU)
- Servo Motor



Power Module

Supplied a stable 3.3 volts to the color sensors, MCU, and Servo Motor.

Supplies an unregulated power source to the DC motor control directly from the batteries.



Power Module

Main requirement that needed measurement was the power supplied to the MCU and color sensors

Input (Volts)	Output from Power Module (Volts)
4.5	3.279
5	3.282
5.5	3.285
6	3.288
6.5	3.292
7	3.295

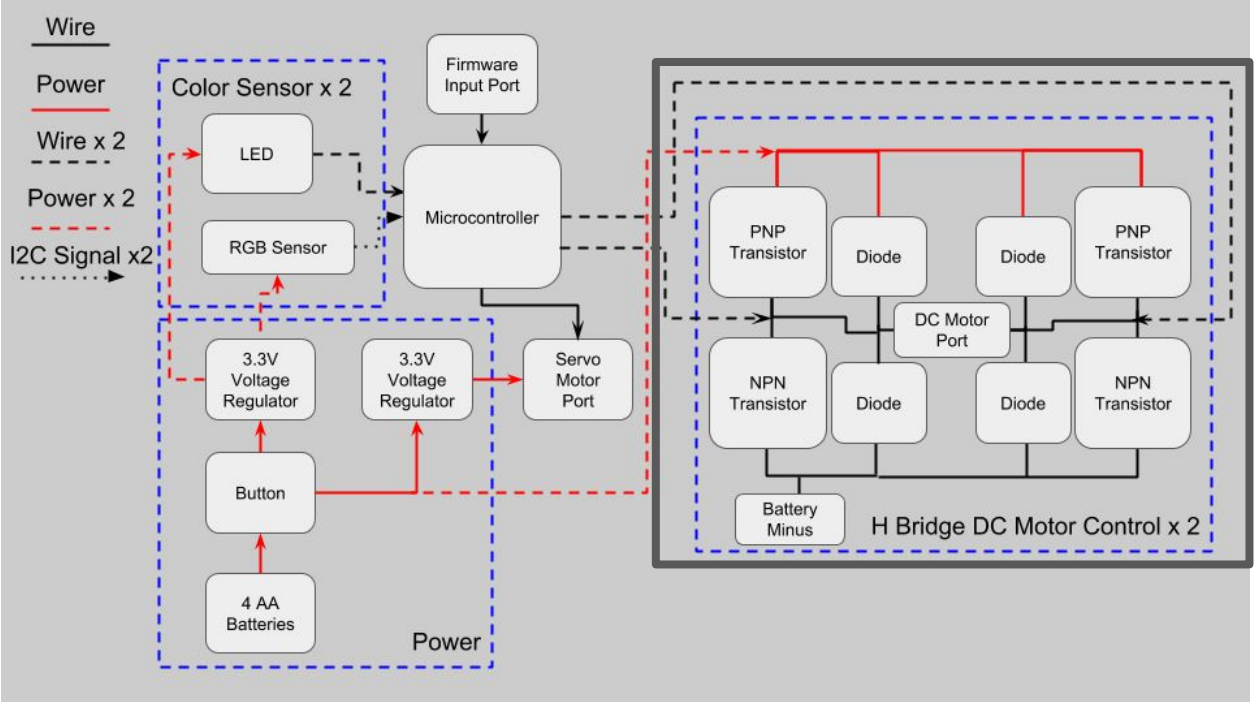
The power module would directly provide ~4.8-6 volts of unregulated power to the DC motor control

DC Motor Control

Composed of two H-Bridge circuits.

Power to the motors is supplied directly from the batteries.

Bias voltage to transistors is provided from the MCU at 3.3 volts.



DC Motor Control

Running both DC Motors takes roughly 230mA of current

IN-1 is by the Positive Lead of the Motor

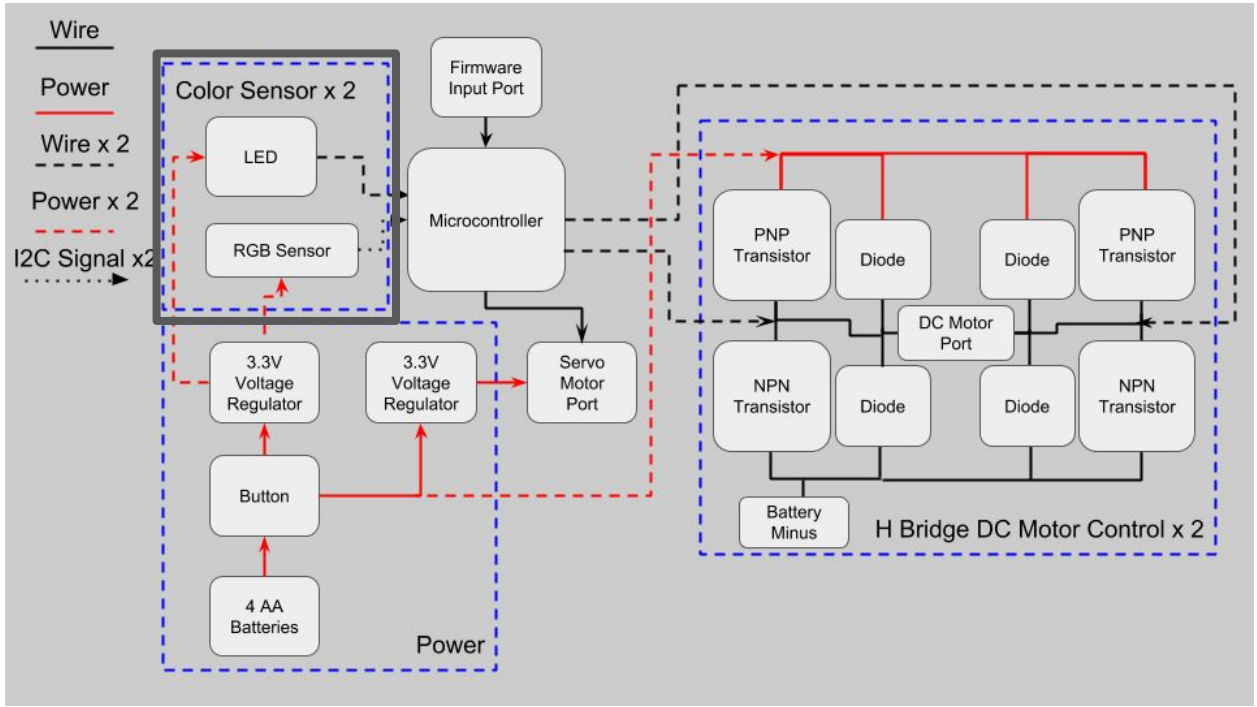
IN-2 is by the Negative Lead of the Motor

IN-1	IN-2	Motor Flow
Low	Low	Stop
High	Low	Clockwise
Low	High	Counter Clockwise

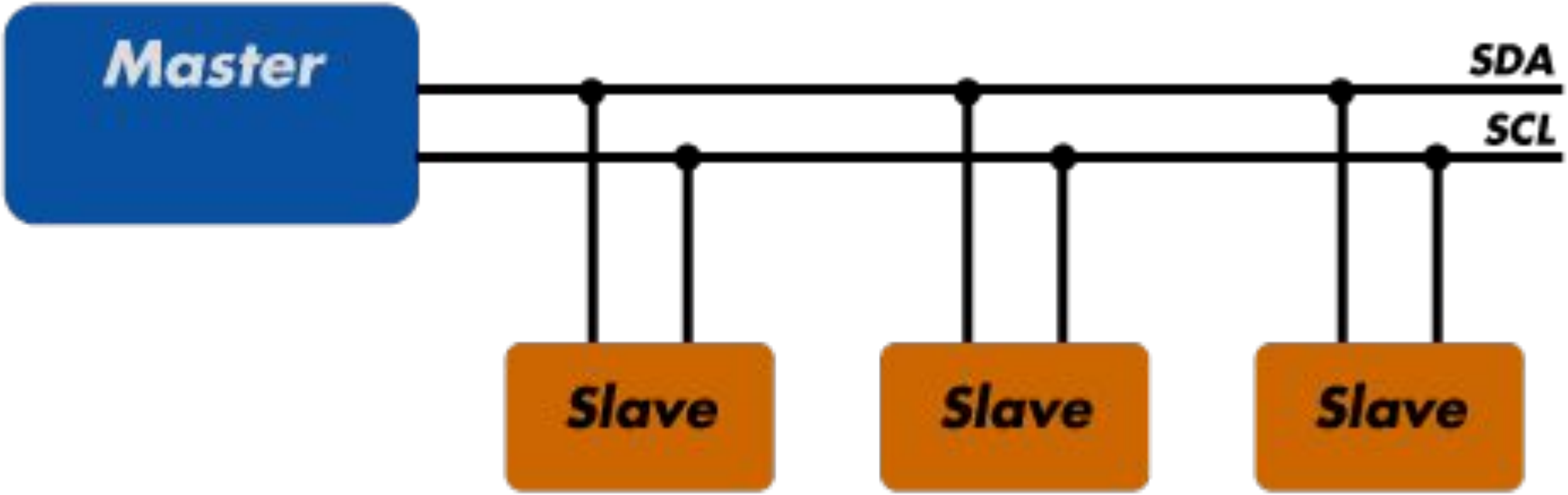
Color Sensors

Detect color in an area illuminated by an LED.

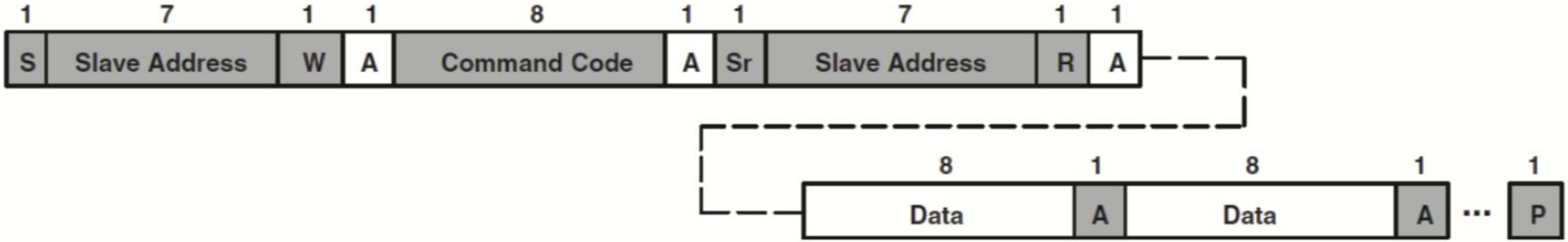
Data is sent through an Inter-integrated Circuit (I2C) interface to the MCU



Inter-integrated Circuit (I2C) Communication Interface

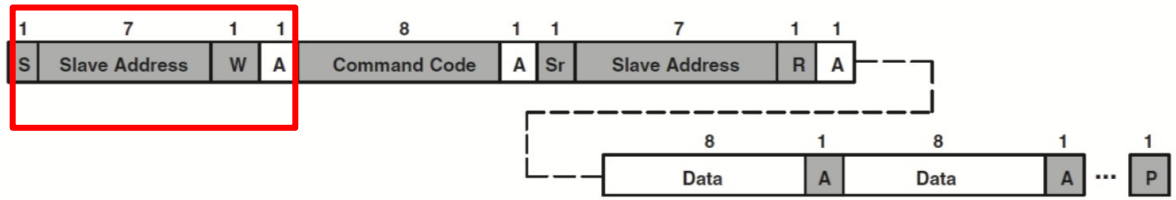


Inter-integrated Circuit (I2C) Communication Interface



I²C Read Protocol — Combined Format

Inter-integrated Circuit (I2C) Communication Interface



I²C Read Protocol — Combined Format

Start + Address + Writing bit

SDA

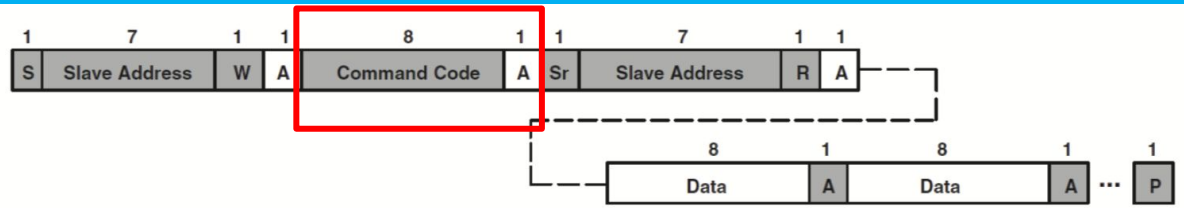
Microcontroller

Color Sensor

SCL

Acknowledge

Inter-integrated Circuit (I2C) Communication Interface



I²C Read Protocol — Combined Format

Register Address

Microcontroller

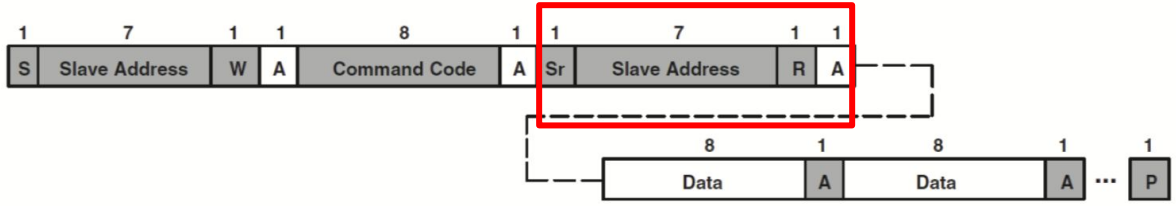
SDA

SCL

Color Sensor

Acknowledge

Inter-integrated Circuit (I2C) Communication Interface



I²C Read Protocol — Combined Format

Restart + Address + Reading bit

Microcontroller

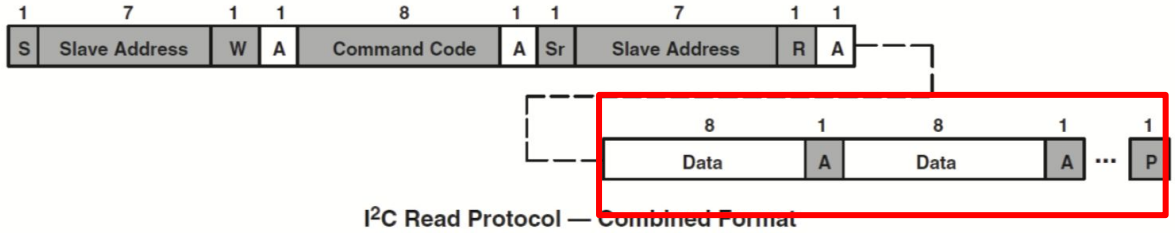
SDA

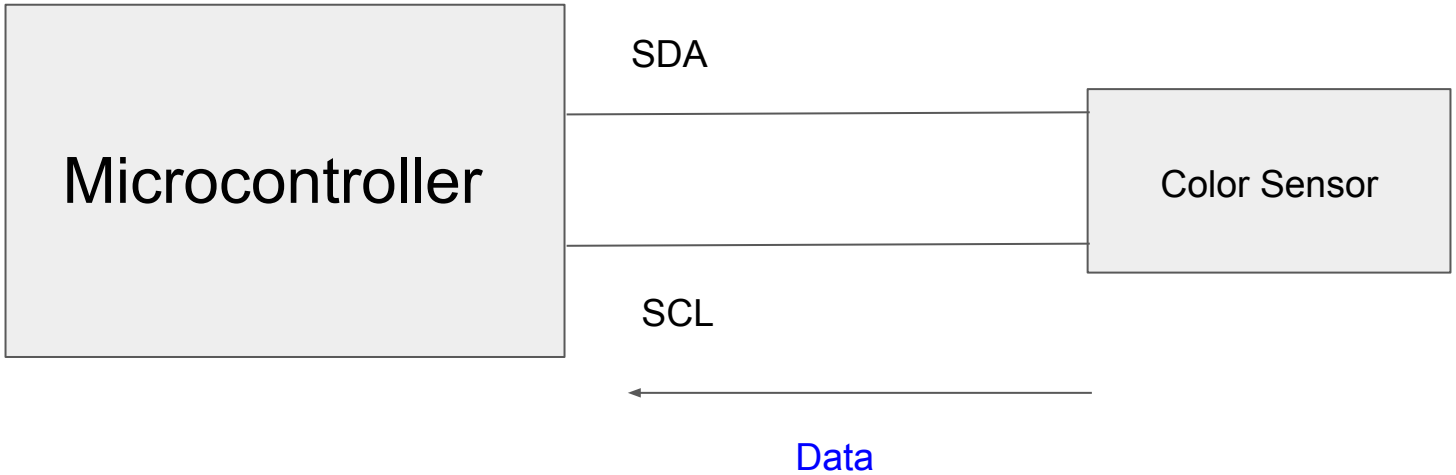
Color Sensor

SCL

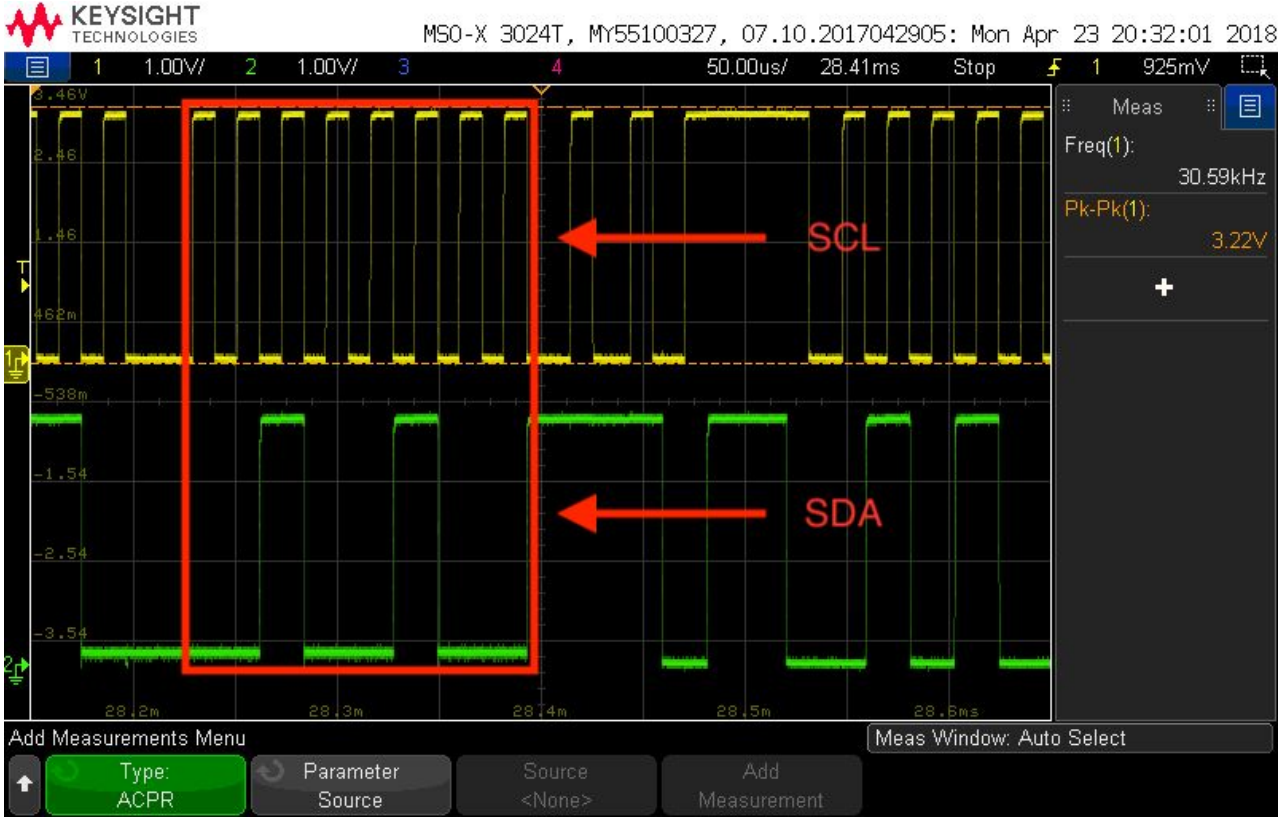
Acknowledge

Inter-integrated Circuit (I2C) Communication Interface





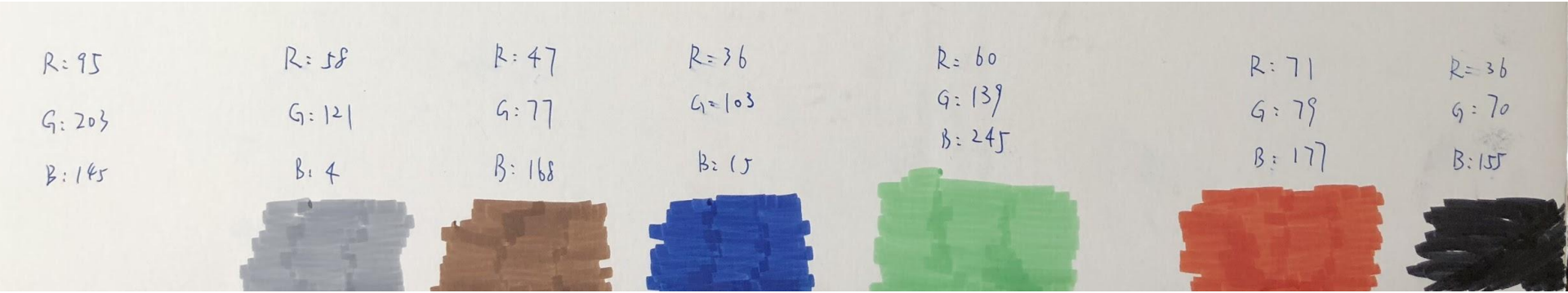
Data Reading Example



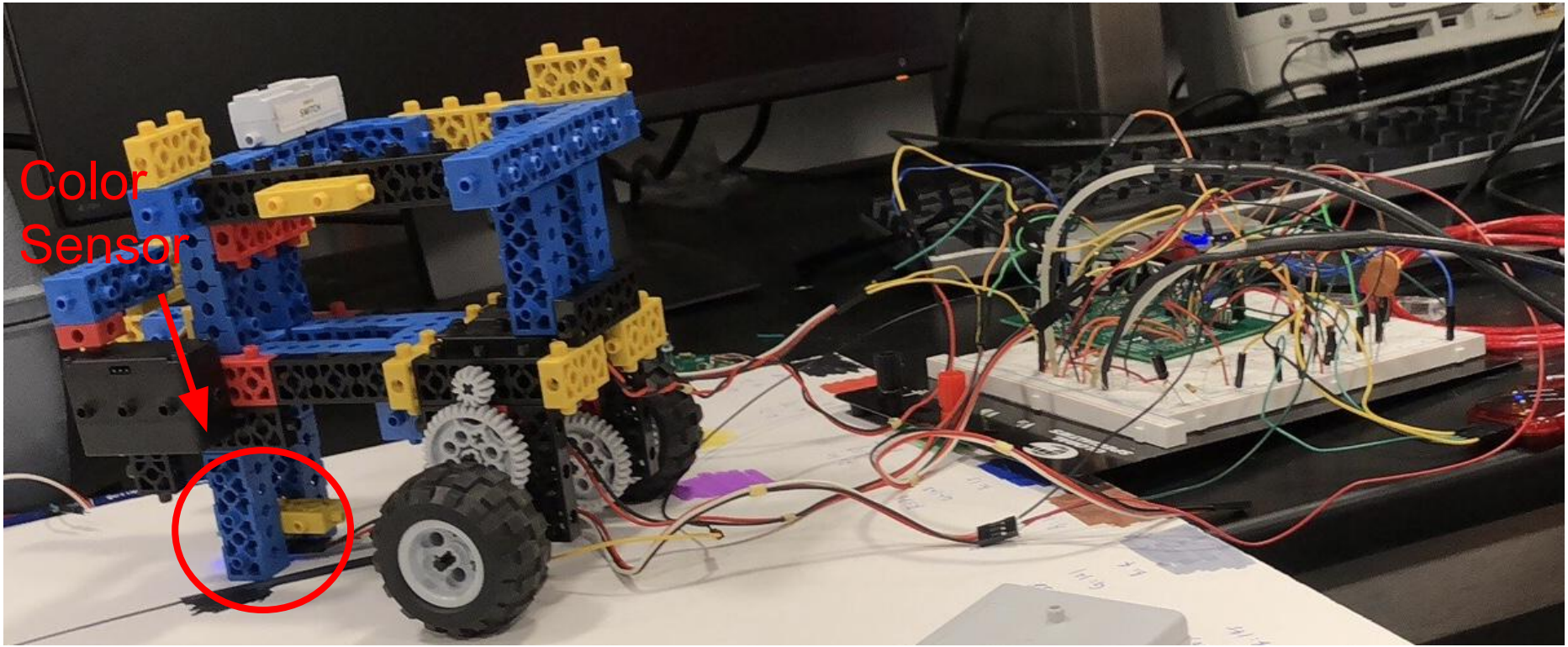
- SCL is the synchronous clock.
- SDA is the real data.
- At each pulse, SDA is read as a binary number.

- 00100100 (36)
- R reading for color blue

Real Color Reading



Physical Design



MCU Color Detector Example

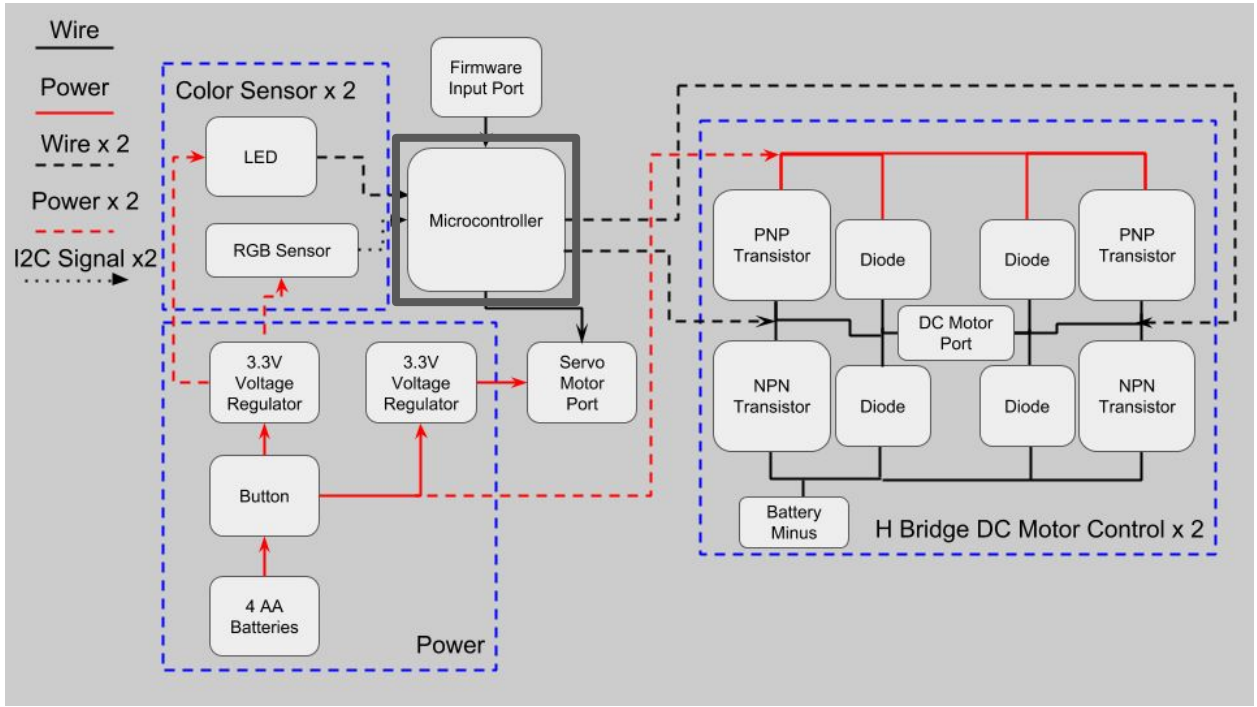
After tuning:

```
int sensor1green() {
    //green detection code here
    unsigned char R = ReadByte_I2C(0x52, 0x80|0x16);
    unsigned char G = ReadByte_I2C(0x52, 0x80|0x18);
    unsigned char B = ReadByte_I2C(0x52, 0x80|0x1A);
    if (R > 0x28 && R < 0x50){
        if (G > 0x6E && G < 0xA0){
            if (B > 0xC8 && B< 0xff){
                return 1;
            }
        }
    }
    return 0;//should be changed to depend on whether
}
```

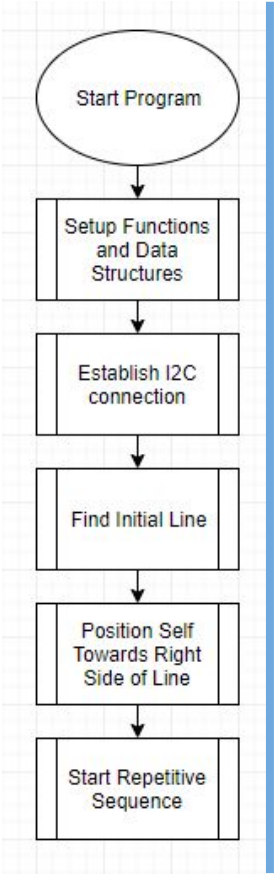
Microcontroller

The MCU is powered off of 3.3 volts from the power module.

It interprets the data received from the color sensors to make decisions on how to control the three motors.



MCU Program Flow



Initializes Functions and Creates Code Word Array

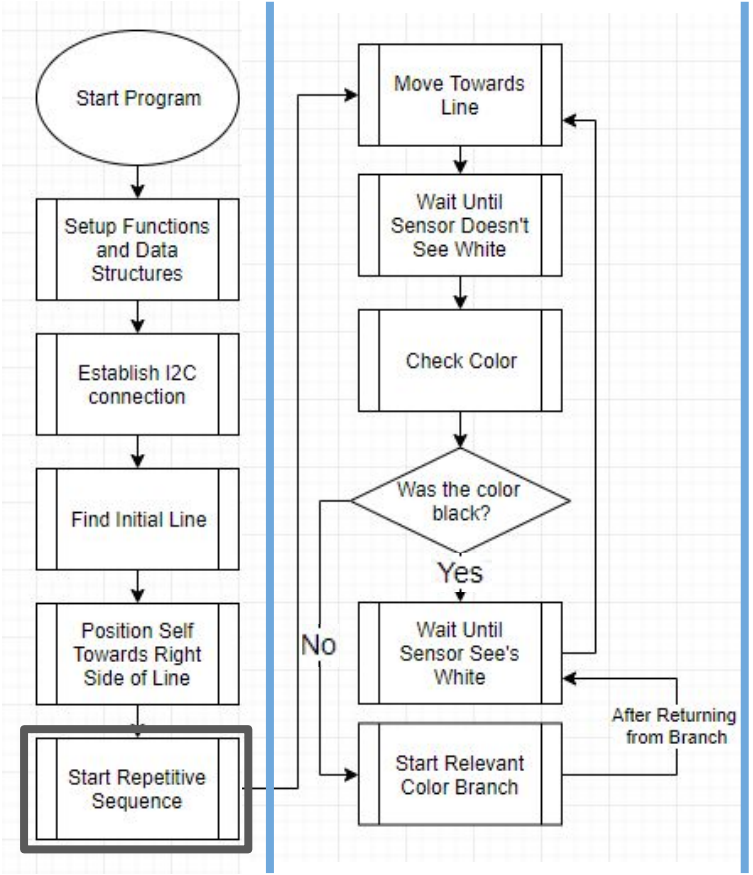
Creates communication between sensor and MCU

Needs to find line to properly position itself

The Robot needs to start to the right of the Line

Sequence of Line Following and Color Detection

MCU Program Flow



Moves Left or Right to Locate Line

Not seeing a background is the Line

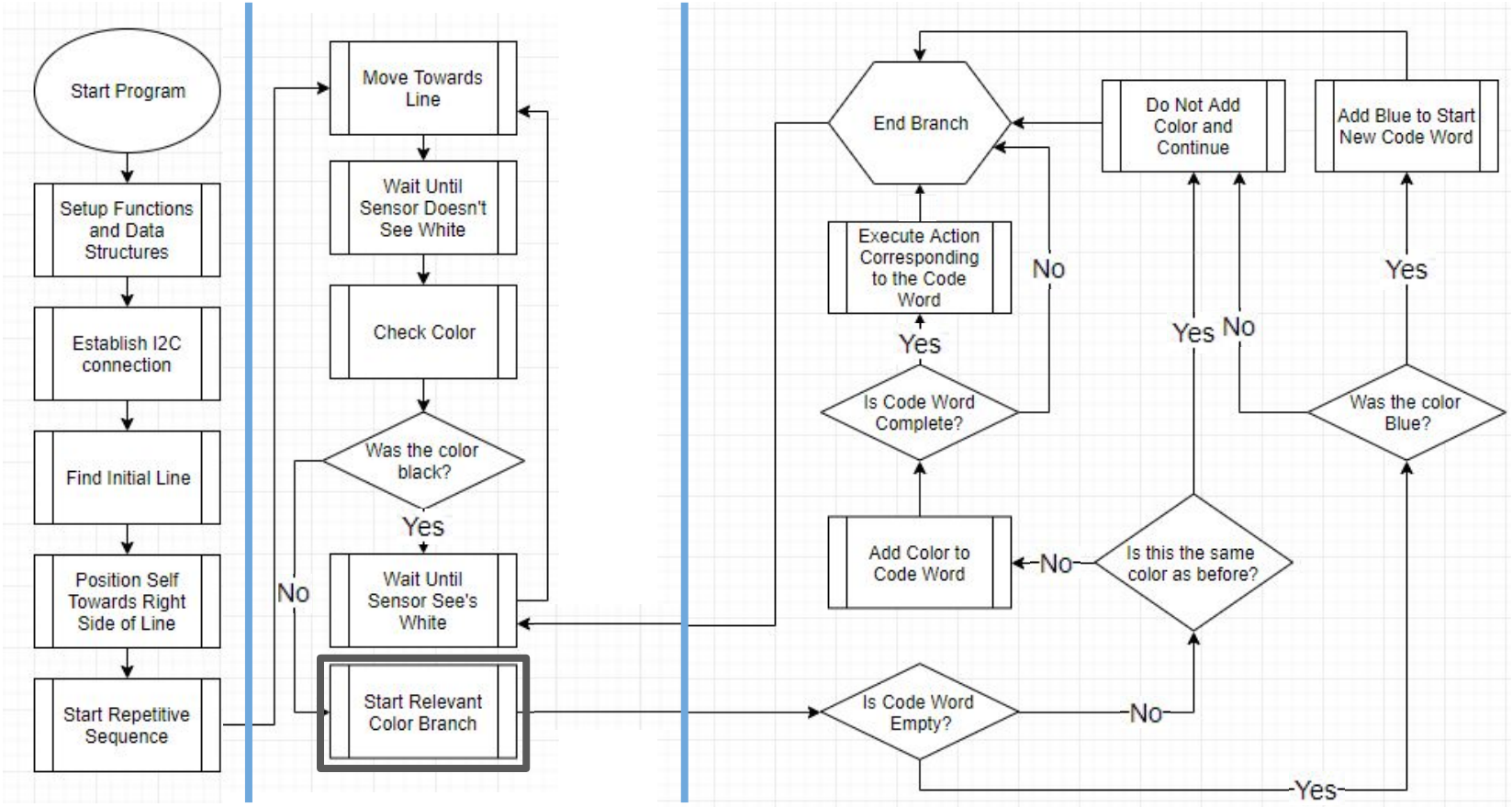
Sensor actively checks the color

Black is the normal line, other colors are code

Continues till the background is seen again

Execution varies depending on the color seen

MCU Program Flow



MCU Program Flow

One of the following instructions would execute before resuming

Stop Moving	=	Blue	Red	Green	Red
Turn Around	=	Blue	Red	Green	Brown
Servo 0	=	Blue	Green	Red	Green
Servo 45	=	Blue	Green	Red	Brown
Servo 90	=	Blue	Green	Brown	Red
Servo 135	=	Blue	Brown	Green	Brown
Servo 180	=	Blue	Brown	Green	Red

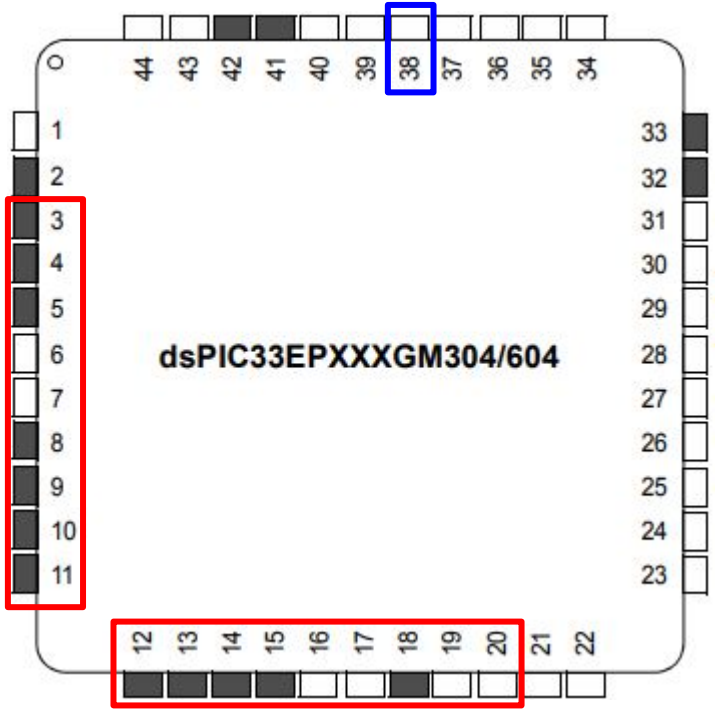
Challenges and Troubleshooting

PCB pads (QFN) for the MCU did not match the connection of the pads of the MCU itself (TQFN).

To accomodate for the issue we were able to fit some pads to connection on the PCB.

Pads outlined in RED were able to fit onto the PCB connections.

The pad in BLUE was brute soldered to a wire and connected with other parts.



Challenges and Troubleshooting

Issues with establishing I2C communication between the sensors and the MCU.

- Can not send out reading instruction. SDA is held low.
  - Solved by increasing time delay and wait until SDA becomes high.

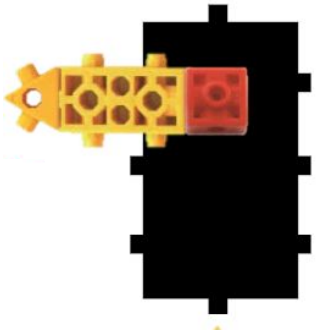
- Response time is too slow.
  - Solved by increasing the ATIME (decrease integration time).

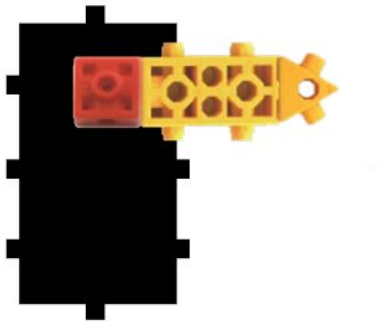
Challenges and Troubleshooting

Servo motors typically require sending a 1-2ms pulse every 20ms to control the angle. Our servo motor did not accurately follow these rules and we had to test for new values.

Normal behavior: 1ms = 0 degrees 2ms=180 degrees

Our behavior: 0.6ms = 0 degrees 2.05ms = 180 degrees





Challenges and Troubleshooting

Encountered issues forming the pulses

The lowest pulsewidth we could form with the built in delay_us function is 1.45 ms

We established a countdown function to execute delay manually

```
void servoPulse(int pulsewidth) {//this should send
    int i = 0;

    while(i < 500) {//the 500 is to ensure enough pu
        int count = 0;
        LATCbits.LATC0 = 1;//set servo pin to high
        while (count < pulsewidth){
        //wait for pulsewidth in microseconds
        count ++;
        }
        count = 0;
        LATCbits.LATC0 = 0;//set servo pin to low
        while (count < 9036-pulsewidth){
            count++;
        }
        i++;
    }
}
```

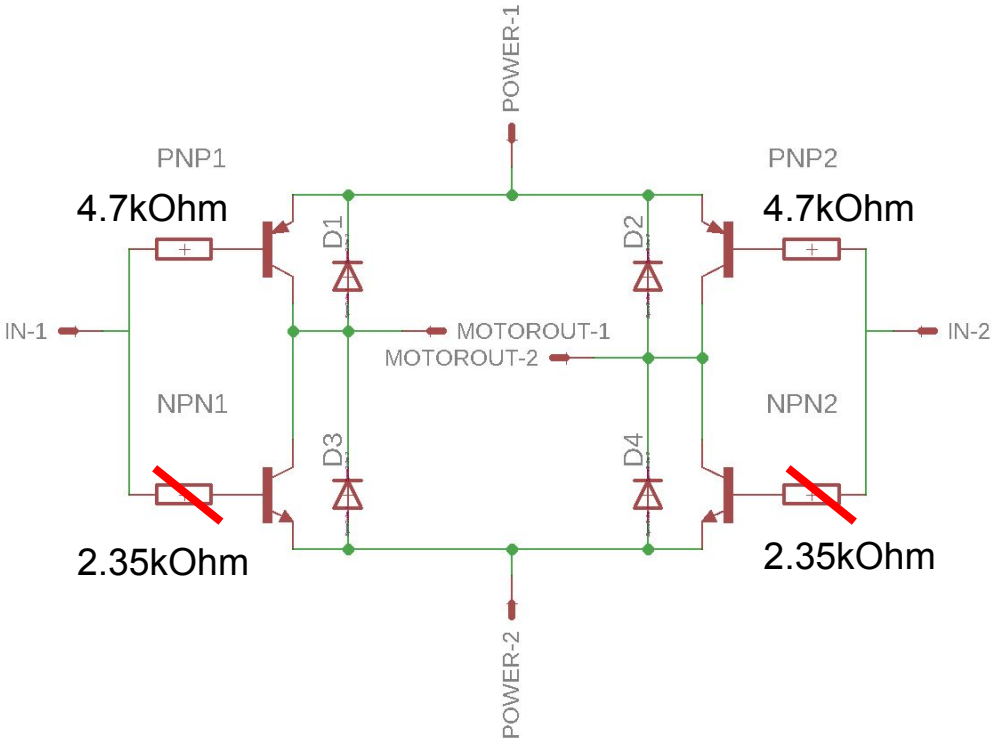
Challenges and Troubleshooting

4.7 kOhm resistors were used at the bias of the transistors

The motors were too weak to drive the robot and drew 120 mA of current

Cutting the resistance on the bias of the NPN transistors in half fixed the issue

The new current draw was 232 mA



Safety and Ethics

Keeping connections as covered as possible is important to ensure people don’t get shocked by the circuit.

Ensuring that the components don’t overheat is important to ensure no part of the board can burn a person.

Future Work

- Scaling down the size to be more compact
- Creating case that interfaces to the blocks we used and also to other popular building platforms like LEGO bricks. As well as adaptors to fit with their electronics.
- Making the electronics adaptable not just in shape but also in electrical compatibility to other electronics such as LEGO motors and common motors used for personal projects

Thank You!

Questions?