Hand-Controlled RC Car

Rohan Punamiya, Saaketh Reddy

ME 4405 Mechatronics - Section A

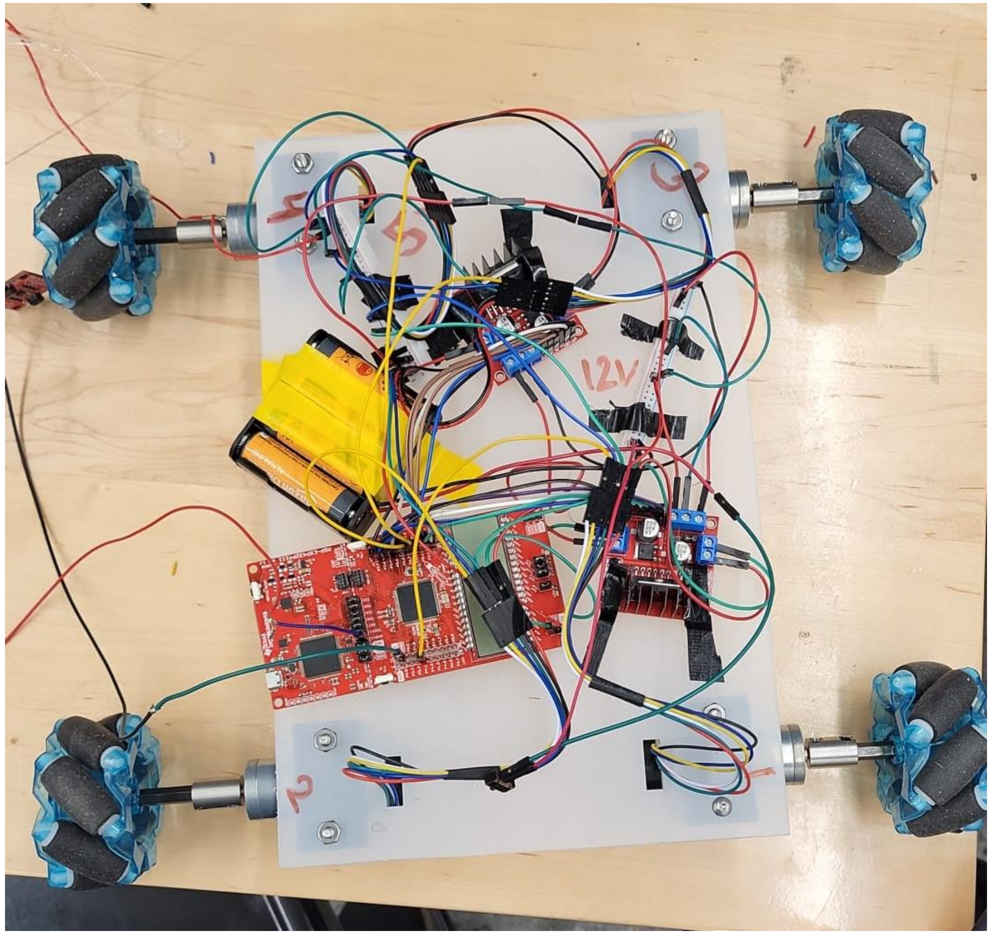
Final Project Presentation

Fall 2023

Georgia Tech

Project Motivation and Performance Goals

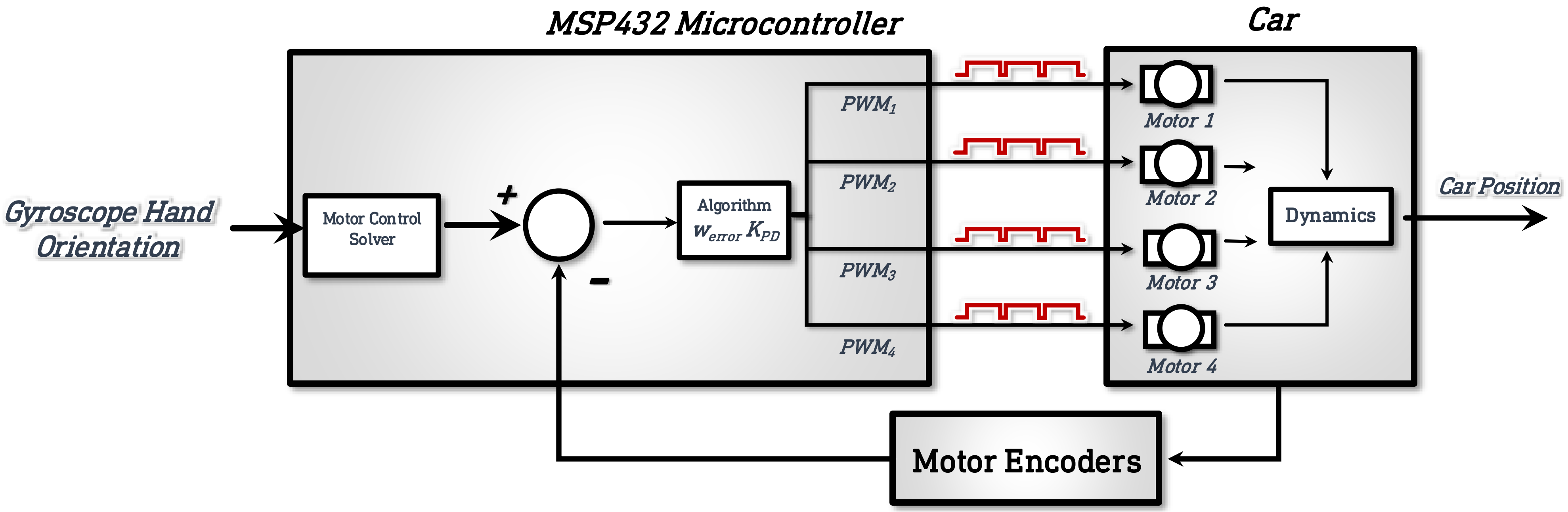
- **Central Idea**: Control the position and speed of an RC car with orientation of hand.
- **Motivations**: Elderly persons find it difficult to pass objects.
- **Goals**: Use the TI MSP432 microcontroller, an IMU, motor encoders, and control theory to enable real-time coordination of DC motors with hand orientation.
- **Performance Requirements**:
  - Establish a closed-loop control system for each wheel, incorporating encoders and IMUs to achieve precise control over speed while enduring perturbations.
  - Create a robust mechanical framework with balanced mass properties, designed to maintain stability and structural integrity.



Georgia Tech

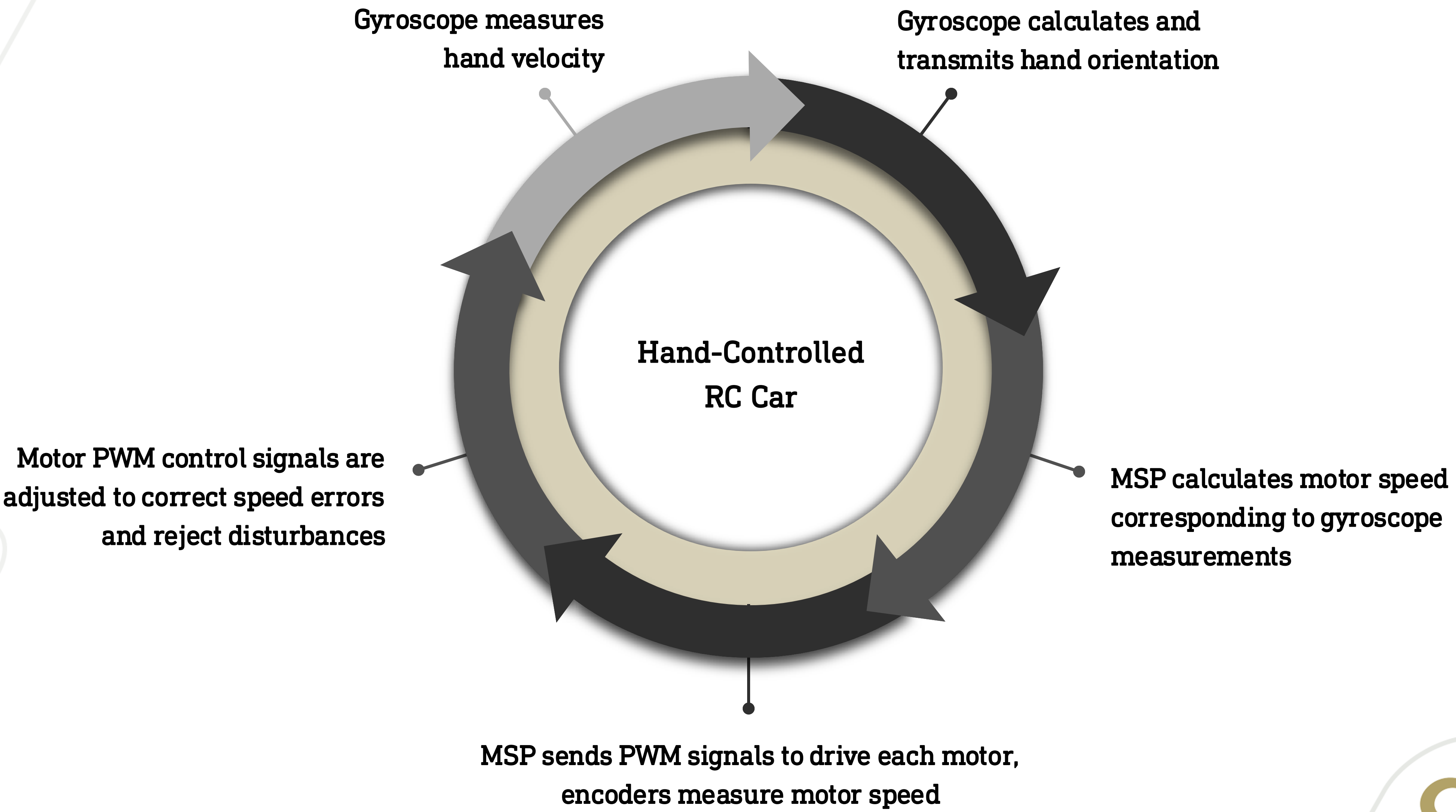
General System Block Diagram and Logic

- **Operating Principle**: Gyroscope in IMU measures hand orientation, MCU feeds motor PWM signals to track hand orientation.
  - Gyroscope sends signals via UART.
  - Encoders on each motor ensure correct speed and disturbance rejection.



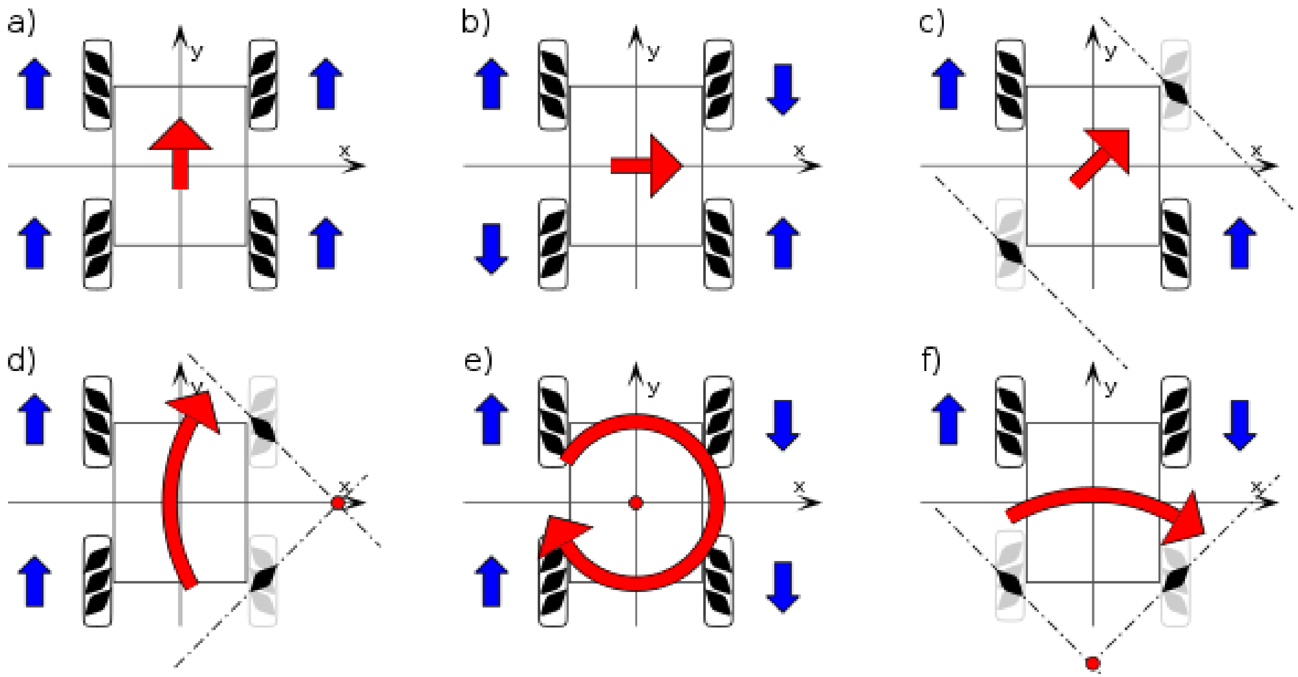
Physical System Diagram

General System Block Diagram and Logic



Mechanical System Design and Components

- **Kinematic Description**
  - Four mecanum wheels determine direction of translation
  - Diagonal wheels have rollers facing same direction.
- **Mechanical Modeling and Estimation**
  - Estimated torque required to move car
    - 0.05 Nm = 0.51kgcm required to move car with maximum acceleration of 1m/s^2
    - Chose factor of safety of 2, implying 2 motors can drive car
- **Actuator Selection**
  - Four BEMONOC 25GA370 DC Encoder Gearmotors
    - Purpose: Power each wheel of the car
    - Specs: 12V, 150 RPM (max), 0.95kg*cm no-speed torque
    - Requirements: 0.51 kg*cm torque output at low speeds



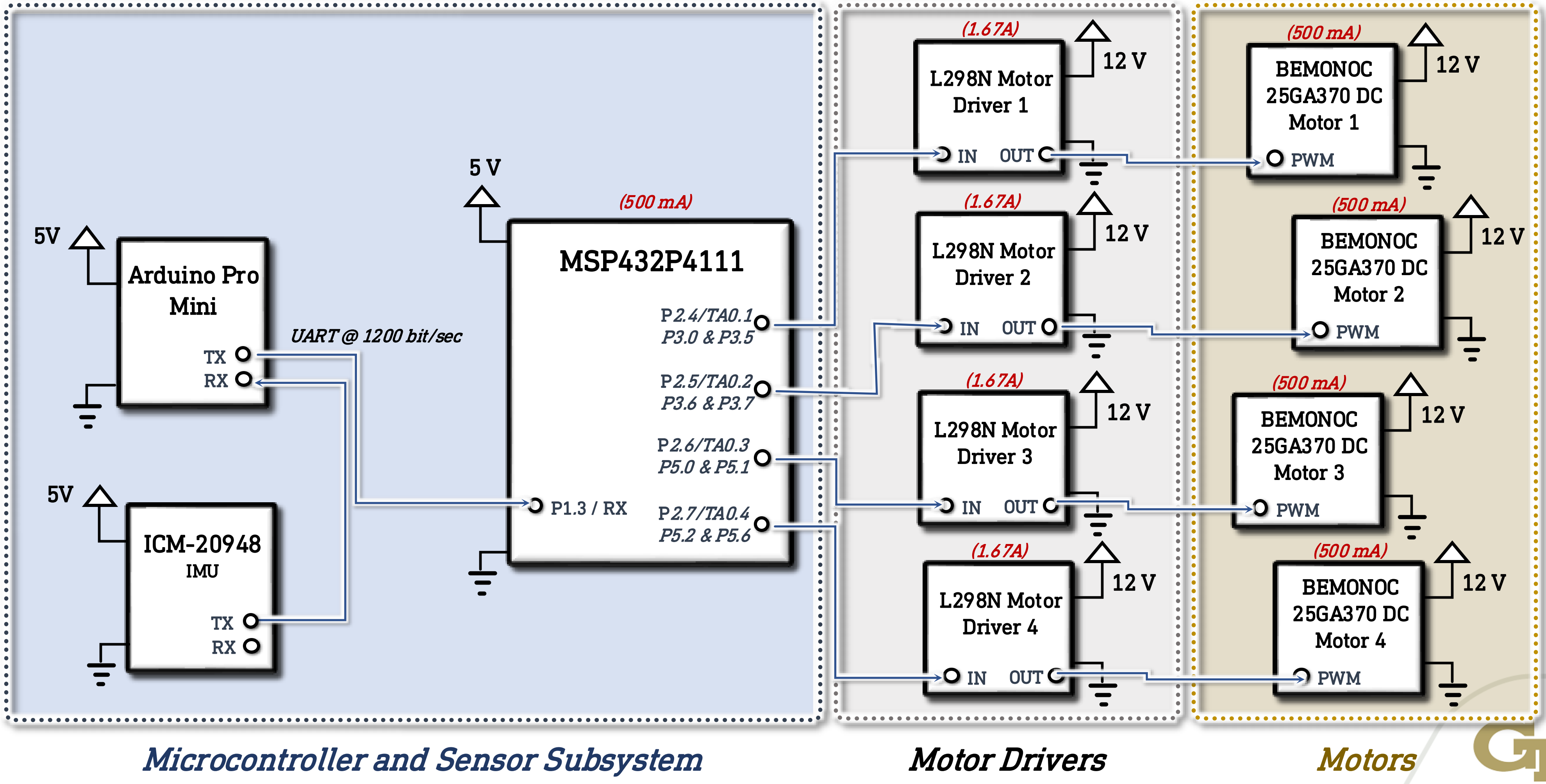
Georgia Tech

Electrical Components and MSP432 Integration

- **General Description of Electrical Design**
  - Subsystems: MCU and Gyroscope, Motor Drivers, Motors
- **Sensor Selection (IMU)**
  - Opted for a 9DOF ICM-20948 gyroscope connected to Arduino Pro Mini, with moving average filter
  - Attempted MCU (nRF52840) containing 6DOF IMU (LSM6DS3TR-C)
    - Filtering techniques such as moving average, complementary, Kalman, mahony, magdewick, low/high-pass filters tried to account for integration drift
- **Power Scheme**
  - To operate for one hour, system requires 71.5W (6W for each motor, 2.5W for MSP, 20W for each motor driver, 5W for the Arduino and gyroscope). This implies a usage of 5541mAh.
  - Used eight 1.5V AA batteries, with a total capacity of 12,760mAh, implying a battery life of 2.3 hours
- **MSP432 Connections**
  - 4 GPIO pins serving as interrupts from encoders pulses
  - 8 GPIO pins serving as highs and lows for motor direction
  - 4 GPIO pins serving as PWM output pins

System Circuit Diagram

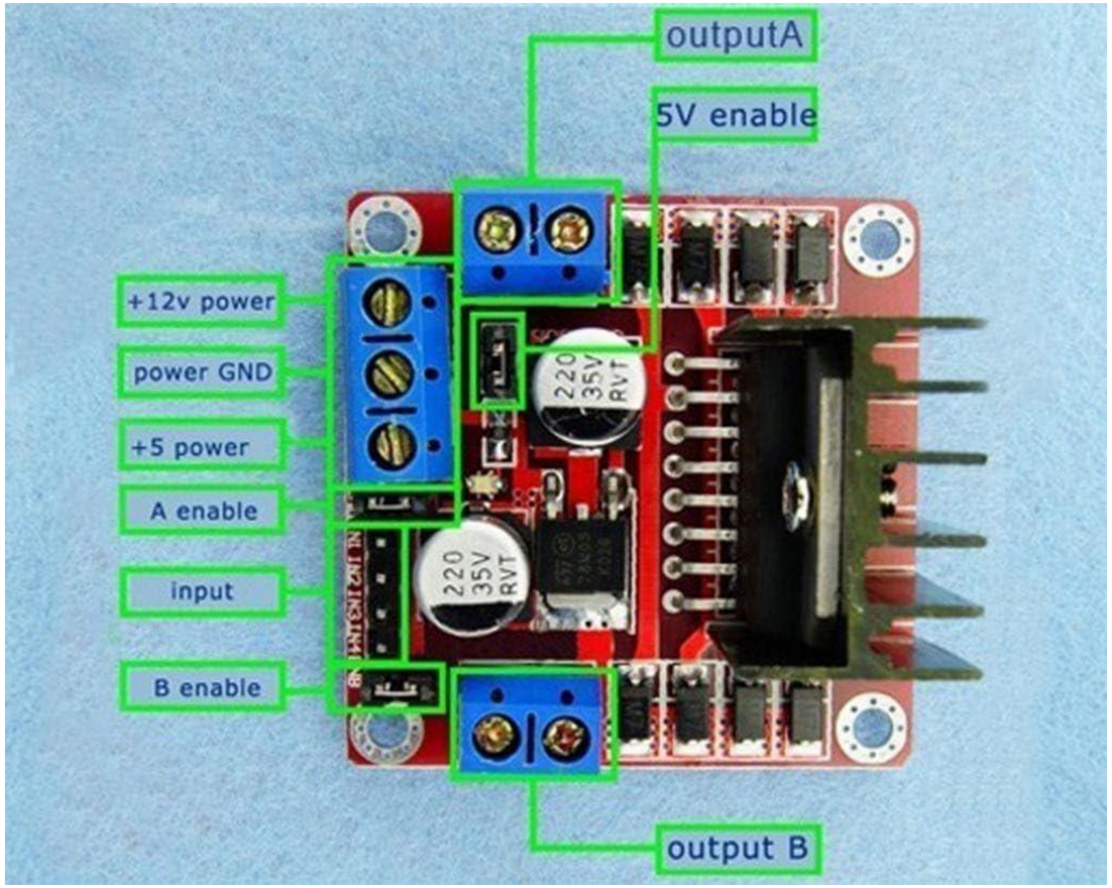
- Systems Approach:



Georgia Tech

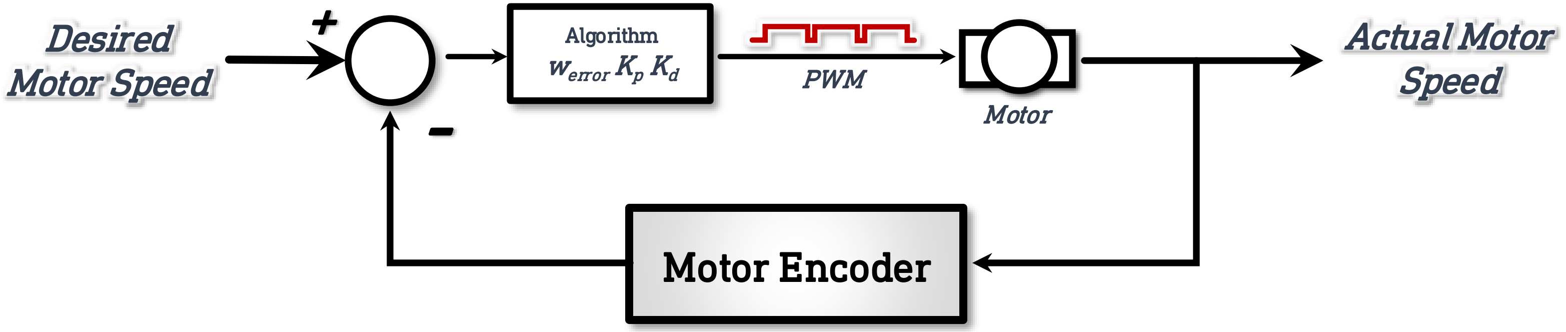
Motor Driver (L298N) Wiring Diagram

- Can operate two motors (output A & B)
- Power GND and +12V to power driver
- Enable pins used for PWM signals
- Inputs 1&2 control direction of motor A
- Inputs 3&4 control direction of motor B
- +5V power used to power MSP



Feedback Control Approach

- **Performance Requirements**
  - **Precision:** ±1 RPM
  - **Speed:** 100ms
  - **Expected Disturbances:** Car hits firm obstacle or has concentrated weight placed on body.
- **Controller Strategy**
  - Diagonal wheels will follow same speed
  - PD Controller (Kp = 5, Kd = 5)
    - Ensures speedy and stable control.



Georgia Tech

Control System Demo

Gyroscope Code Structure

- **Setup**
  - Starts serial communication with MSP
  - Calibrates gyroscope orientation
- **CalibrateGyro()**
  - Measures gyroscope position offset by integrating velocity when IMU is at rest
- **Main Loop**
  - Gyroscope velocity is measured and integrated to find gyroscope orientation
  - Finds absolute position by subtracting calibration offset orientation angles
  - Transmits orientation data via UART to MSP

MSP Code Structure

- **MSP432 Processing Details**
  - Clock operates at 3MHz
  - Implemented UART at 1200 baud rate for minimal data loss
  - Implemented timer module at 50Hz for PWM
  - Gave priority to timer interrupt over encoder interrupts
  - Included nothing in the while loop with so many interrupts involved

```
// 5. Set priority
Interrupt_setPriority(INT_PORT2,4);
Interrupt_setPriority(INT_PORT4,4);
Interrupt_setPriority(INT_PORT5,4);
Interrupt_setPriority(INT_PORT6,4);
Interrupt_setPriority(INT_TA0_0, 0);
```

```
    //    ---------------LOOP-----------

    // Infinite loop
    while(1){

    }


}
```

Georgia Tech

Encoder Interrupts

```
void PORT2_IRQHandler(void){
    pulsecount_wheel3++;
    GPIO_clearInterruptFlag(GPIO_PORT_P2, GPIO_PIN3); // Reset counter
}

void PORT4_IRQHandler(void){
    pulsecount_wheel1++;
    GPIO_clearInterruptFlag(GPIO_PORT_P4, GPIO_PIN1); // Reset counter
}

void PORT5_IRQHandler(void){
    pulsecount_wheel4++;
    GPIO_clearInterruptFlag(GPIO_PORT_P5, GPIO_PIN7); // Reset counter
}

void PORT6_IRQHandler(void){
    pulsecount_wheel2++;
    GPIO_clearInterruptFlag(GPIO_PORT_P6, GPIO_PIN7); // Reset counter
}
```

- Made encoder interrupts as short as possible because we have 2000+ interrupts per second

Gyroscope Interrupt

- Receive the IMU readings as chars (convienient for UART) and then stringing them and converting them into integers

Challenges

- Print statement in debugging was a major challenge as it was slowing the system down resulting in data loss

```
void EUSCIA0_IRQHandler(void){

    // Read the data from the RX buffer
    receivedBuffer[count] = EUSCI_A_UART_receiveData(EUSCI_A0_BASE);


    if(receivedBuffer[count] == '\n'){
        receivedBuffer[count] = '\0';

        sscanf(receivedBuffer, "%d", &Gyro);
        count = 0;



    } else {
        count++;
    }
```

Georgia Tech

Implementation of PD controller

- Hard to map response rate as computer could not be connected to system as it was running at full power
    - 12V power supply in the system blew up motherboard of laptop
- Had to tune PD controller blindly

```
dt = 0.1;

p_gain = 5;
d_gain = 5;
```

```
prev_PWM1_diff = PWM1_diff;

PWM1_diff = PWM1_target - PWM1_real;

if ((PWM1_diff + PWM1_target) > 5000)
{
    PWM1_target = PWM1_target + (PWM1_diff * p_gain) +  d_gain * (PWM1_diff - prev_PWM1_diff) * dt;
}
```

Acquiring Real Speed Values

- Converting encoder values to speed.
- Here PWM is analogous to speed, so essentially we get real speed.

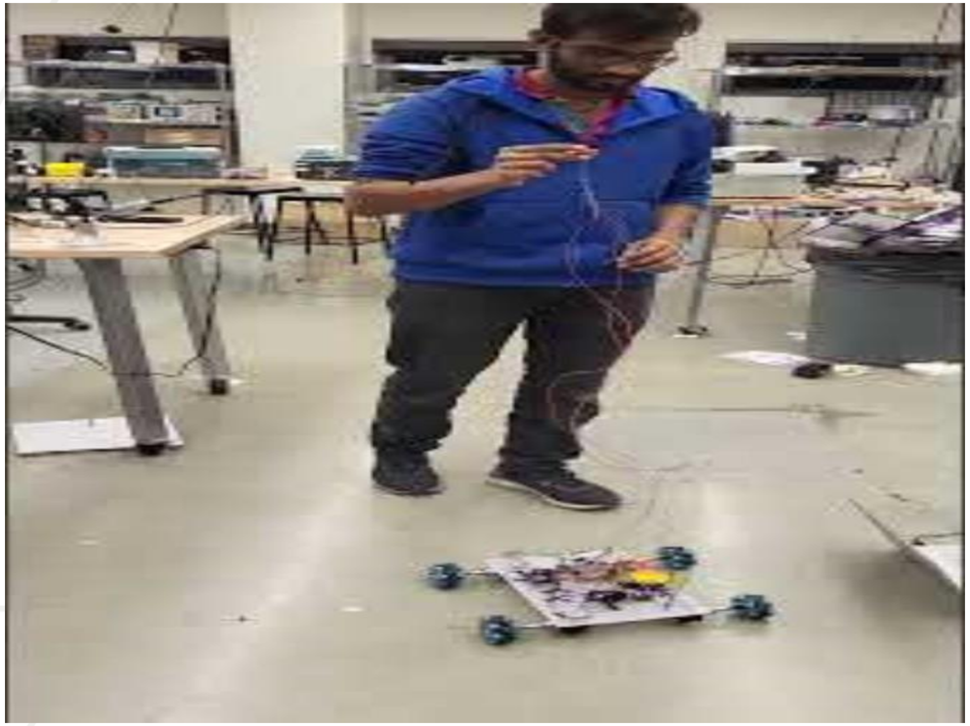
```
PWM1_real = PWM1_target * (pulsecount_wheel1 / (410 *multiplier / second_divider));
PWM2_real = PWM2_target * (pulsecount_wheel2 / (410 *multiplier / second_divider));
PWM3_real = PWM3_target * (pulsecount_wheel3 / (410 *multiplier / second_divider));
PWM4_real = PWM4_target * (pulsecount_wheel4 / (410 *multiplier / second_divider));
```

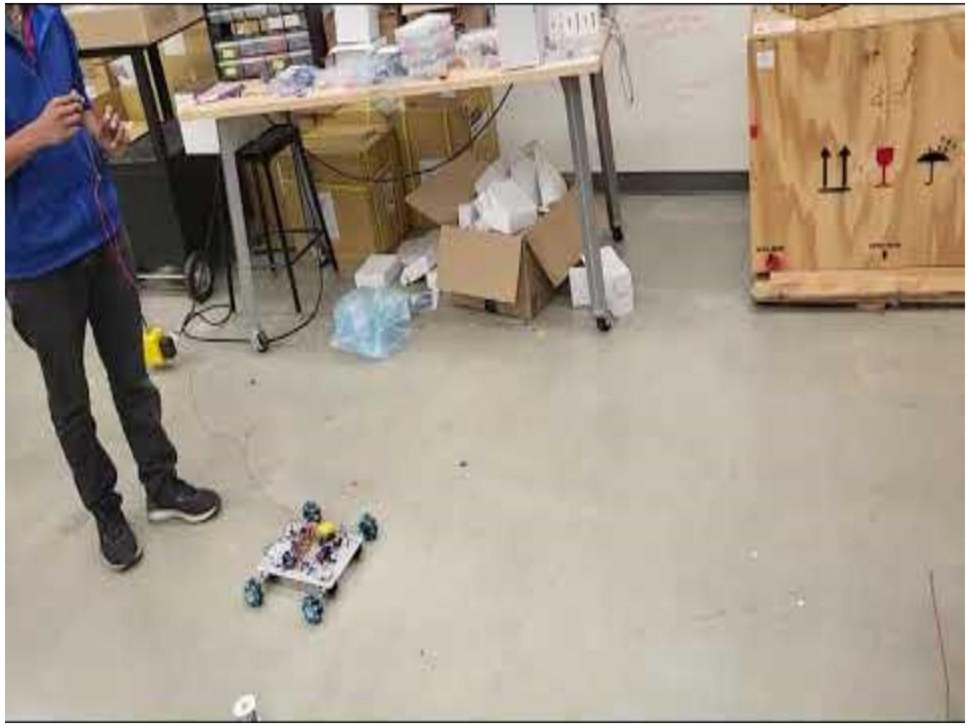
Georgia Tech

Results of Mechatronic System Evaluation

- **Description of evaluation process**
  - Car travelling forwards and backwards is tested
  - Success of mechatronics system measured by position error in path following
  - Car translation is tested with 2kg weight placed at the back
  - Success of control system measured time error in path following with weights
- **Analysis of Results**
  - System is responsive within 100ms, and follows intended path (with maximum offset of 0.05m for every 1m travelled). This 5% error may be due to mechanical inconsistencies or with encoder resolution of 7.5 degrees.
  - Without any concentrated weight, the car travelled a distance of 2m in 8.71s. With a 2kg weight, the car travelled this distance in 8.75s.

Demo Videos





Challenges and Potential Improvements

- **Challenges and Solutions**
  - Store-bought RC car was too compact, final RC car had to be designed, fabricated, and assembled from scratch (besides the wheels).
  - The gyroscope was difficult to get clean readings from, ended up using a moving average filter
  - After initially attempting to use PID controller, ended up using a PD controller due to instability. Required lots of trial and error to get adequate gains.
- **Potential Improvements**:
  - Allow remote communication via BLE signals from gyroscope (or even phone).
  - Add greater mechanical suspension to reduce ground perturbations.
  - Add a camera and use OpenCV to allow obstacle avoidance.