

Running Pace Assistant



Group 5

David Creger, Gaurav Gunupati, Ben Chang

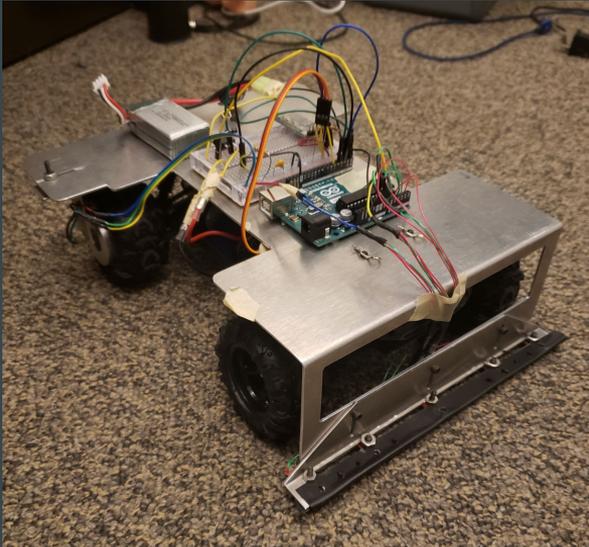
Intro

- Maintaining a constant speed during a distance race leads to the fastest times
- Help runners develop muscle memory for their desired pace
- Provide instantaneous feedback that other devices don't



Overview

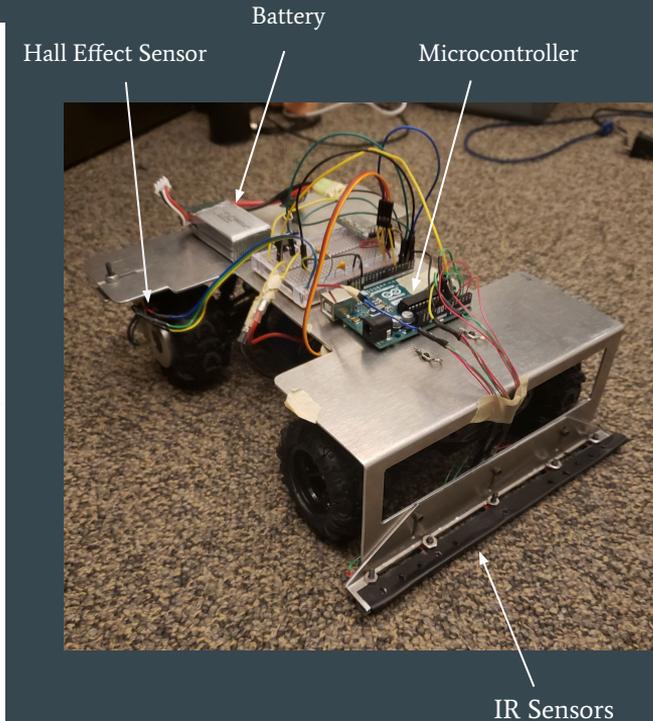
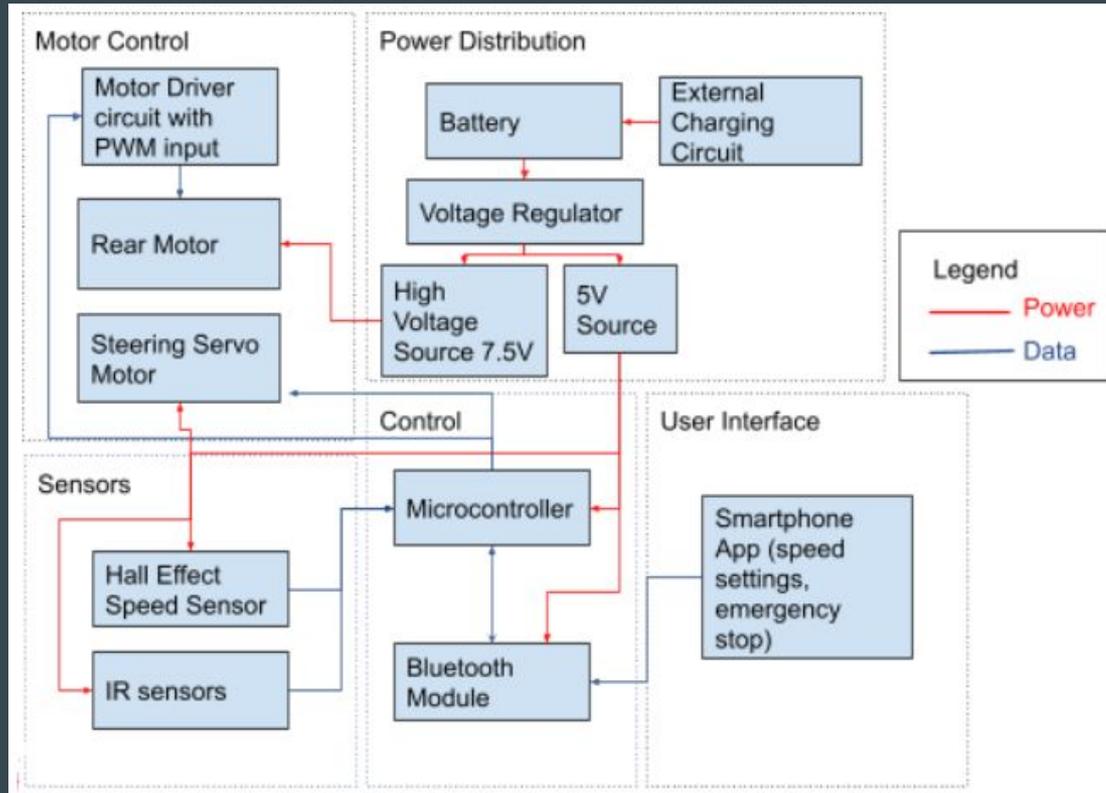
- Pace assistant runs at a constant speed on a standard running track
- Follows lane line around track using IR sensors
- Paired with a smartphone app for easy remote operation.



High Level Requirements

- The robot must have adjustable speed ranging from 5 to 10 mph, and be able to operate for at least 30 minutes at 6mph.
- The robot must follow all typical Olympic track lane markers at all times using IR sensors.
- The smartphone app must have a display showing set speed, distance travelled, and time elapsed. Distance, pace, and time must each be correctly displayed with an allowable error of 5%.

System Overview



MIT App Creator



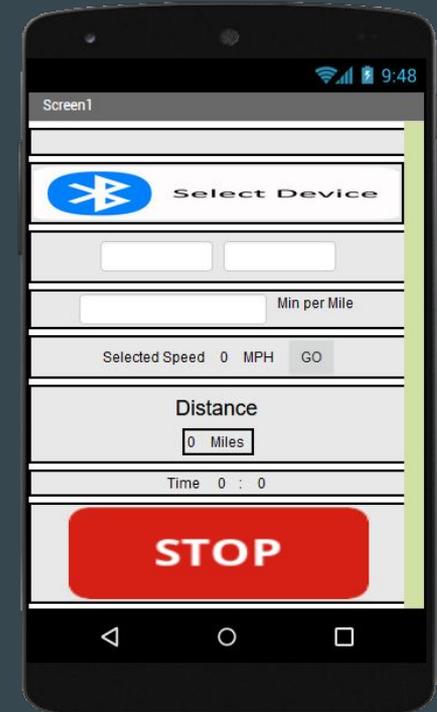
- Similar to scratch, allows for modular programming of individual components
- <https://appinventor.mit.edu/>
- Block based tools

A screenshot of the MIT App Inventor code editor showing several event-driven code blocks. The blocks are color-coded and connected by lines. The visible code includes:

- When Clicked (Click):** Checks if 'global start' is false. If true, it sets 'global timer' to 0, 'set_time' to 'Text' * 60, and 'global start' to true. It then calls 'BluetoothClient1' SendText with 'Selected_Speed' and '\n'.
- When Selected Bluetooth (BeforePicking):** Sets 'Select_Bluetooth' Elements to 'BluetoothClient1' AddressesAndNames.
- When Selected Bluetooth (AfterPicking):** Sets 'Select_Bluetooth' Selection to 'BluetoothClient1' Connect address 'Select_Bluetooth' Selection. It also sets 'Select_Bluetooth' Text to 'Device Connected'.
- When Emergency_Stop (Click):** Calls 'BluetoothClient1' SendText with '0' and '\n', and sets 'global start' to false.
- When Speed_Select_Text (LostFocus):** Checks if 'Speed_Select_Text' Text is not empty and not 0. If true, it sets 'Min_to_mile' Text to 60 / 'Speed_Select_Text' Text, and 'Selected_Speed' Text to 'Speed_Select_Text' Text. Otherwise, it sets 'Min_to_mile' Text to 0.
- When Min_to_mile (LostFocus):** Checks if 'Min_to_mile' Text is not empty and not 0. If true, it sets 'Speed_Select_Text' Text to 60 / 'Min_to_mile' Text, and 'Selected_Speed' Text to 'Speed_Select_Text' Text. Otherwise, it sets 'Speed_Select_Text' Text to 0.
- When Clicked (Clock1 - Timer):** Checks if 'global start' is true. If true, it sets 'Current_Speed' Text to 'BluetoothClient1' ReceiveText numberOfBytes, and calls 'BluetoothClient1' BytesAvailableToReceive. It then checks if 'global timer' is 0. If true, it calls 'BluetoothClient1' SendText with '0' and '\n', and sets 'global start' to false. If false, it increments 'global timer' by 1, sets 'Min' Text to quotient of 'global timer' + 60, and 'Sec' Text to remainder of 'global timer' + 60.
- Initialize global start to false** and **initialize global timer to 0** are shown as separate initialization blocks.

User Interface

- Allows the user to either input speed or pace
- Converts it both ways
- Allows user to set a time
- Go starts the RC car, and automatically stops
- Shows the total distance calculated by the RC car



Requirement	Verification
1. Sends a stop signal to the car when the button is pressed.	A. Measure time elapsed against a stopwatch.
2. Accurately sends the required speed value.	B. Read value that is sent on the microcontroller end
3. Accurately displays time elapsed.	



3:27
Green7

50%



Select Device

6

10

10

per Mile

0 Miles

Time 0:0

STOP



Speed Control



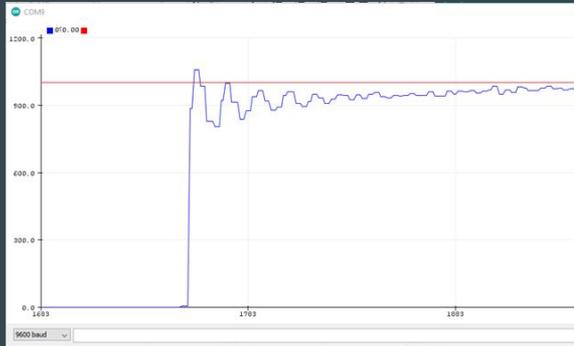
Circumference = 210mm

$$RPM = \frac{X \text{ mi}}{\text{hr}} * \frac{1 \text{ hr}}{60 \text{ min}} * \frac{1,609E6 \text{ mm}}{\text{mile}} * \frac{1 \text{ rotation}}{210 \text{ mm}}$$

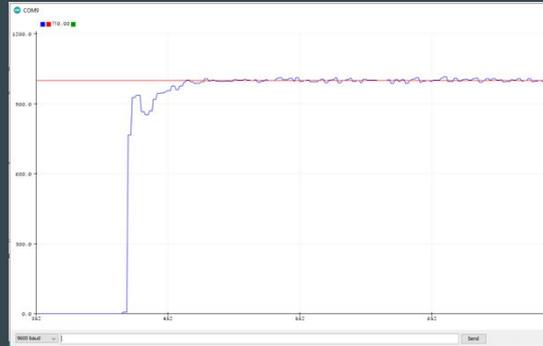
*Calculation of Desired RPM from mph

- Hall Effect Sensor
- PID Controller
- Requirement: Car drives at speed within 5% of target value
- Verification: Time with stopwatch how long it takes to travel 100m

```
float kp = 0.75;
float ki = 0.01;
float kd = 0.00001;
```



```
float kp = 0.5;
float ki = 0.1;
float kd = 0;
```



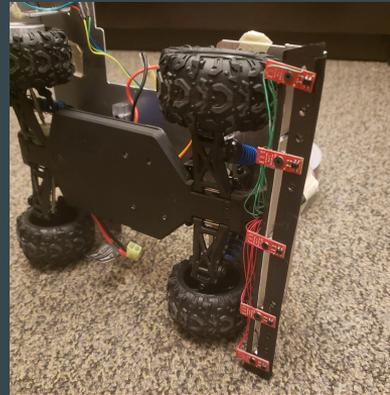
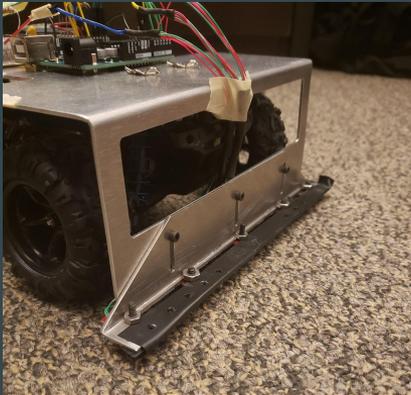
Speed	Calculated Time	Actual Time	Percent Error
3.5 mph	22.37 sec	23.05 sec	3.0%
4 mph	19.57 sec	20.06 sec	2.5%

*Tests conducted on 35m line

Steering Control

- 5 IR sensors
- Lookup table for servo motor position
- Requirement: The robot must follow all typical Olympic track lane markers at all times
- Verification: Tape test in ECEB

IR SENSOR VALUES	10000	11000	01000	EVERYTHING ELSE	00010	00011	00001
SERVO POSITION	132	133	134	136	138	139	140





Microcontroller

IR sensors and Steering Code

```
Left2 = analogRead(2); //reads IR sensor values
Left1 = analogRead(1);
Center = analogRead(4);
Right1 = analogRead(3);
Right2 = analogRead(0);

if (Left1 < Line && Left2 > Line && Right1 > Line && Right2 > Line){ //Turns wheels
  position = 139;
}
else if (Left1 < Line && Left2 < Line && Right1 > Line && Right2 > Line && Center > Line){
  position = 140;
}
else if (Left1 > Line && Left2 < Line && Right1 > Line && Right2 > Line && Center > Line){
  position = 141;
}

else if (Left1 > Line && Left2 > Line && Right1 < Line && Right2 > Line){ //Turns wheels
  position = 134;
}
else if (Left1 > Line && Left2 > Line && Right1 < Line && Right2 < Line && Center > Line){
  position = 133;
}
else if (Left1 > Line && Left2 > Line && Right1 > Line && Right2 < Line && Center > Line){
  position = 132;
}

else{
  position = 136;
}

servo.write(position);
```

Speed Control Code

```
RPM = 6000000 / DeltaT; //calculates RPM from DeltaT

if(micros() - t_prev > 500000){ //sets RPM to 0 if we get no pulses in 0.5 seconds
  RPM = 0;
}

error = RPM_desired - RPM; //calculates error
integ_err = integ_prev + (0.0000001*DeltaT * ((error + error_prev) / 2)); //trapezoidal

if(RPM_desired == 0){ //enforces that integral is 0 when desired rpm is 0 so that the
  integ_err = 0;
}

Dutycycle = kp*error + ki*integ_err + (kd * (error - error_prev) / (DeltaT*0.0000001)) ;

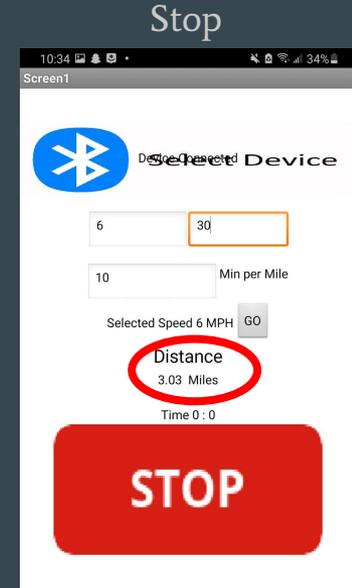
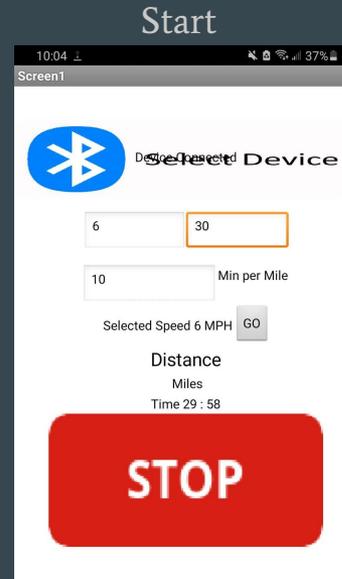
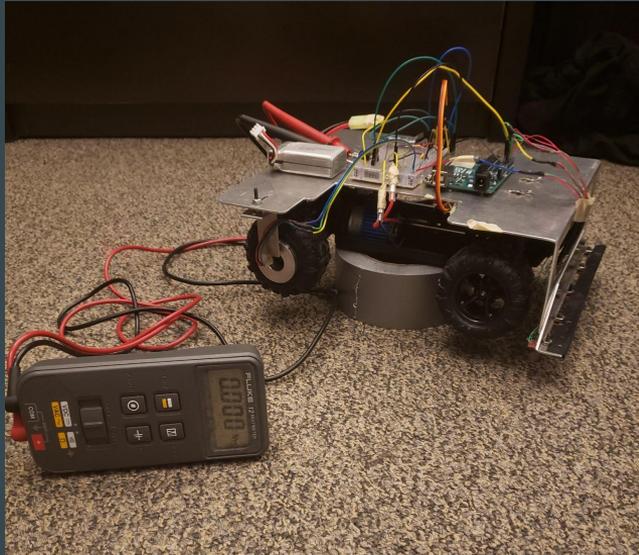
if (Dutycycle > 255){ //Anti-wind up. Caps duty cycle at 255 and maintains error
  Dutycycle = 255;
  integ_err = integ_prev;
}

if (Dutycycle < 0){ //prevents negative duty cycles
  Dutycycle = 0;
}
```

Battery Test

Requirement: Car must be able to drive 3 miles on a single battery charge

Verification: Endurance test on stand



Successes

- The car worked perfectly in the ECEB with breadboards



Challenges

- No track access
- PCB issues
- Steering algorithm is very basic and required a lot of tuning

Conclusion and Future Work

- Problems
 - Fix PCB
 - Add more IR sensors
 - Implement a calibration feature for different colors of track and line markings
 - Fix the overlapping text after selecting a bluetooth device on the app

We hope that with these improvements, we can create a great tool for new and experienced runners alike.

Thank you



ANY QUESTIONS?