GO BREAD: Autotrike

Weston White, Magnus Gu, Kevin Oh, Sarah Kung

Opportunity

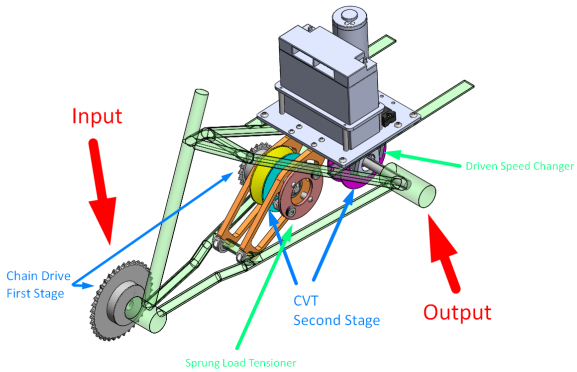
As one of the most simple modes of mechanical transportation, bicycles have always served as one of the most affordable and environmentally friendly ways of traveling. However, bikes can be difficult machines to operate because of the finger dexterity and balance required for riding – especially for users with disabilities. Our team saw this as a key area for improvement. By automatically driving the gear shifting mechanism with a CVT we can remove the finger dexterity requirement while also improving ride smoothness. Furthermore, by augmenting an existing tricycle we remove the need for balancing.

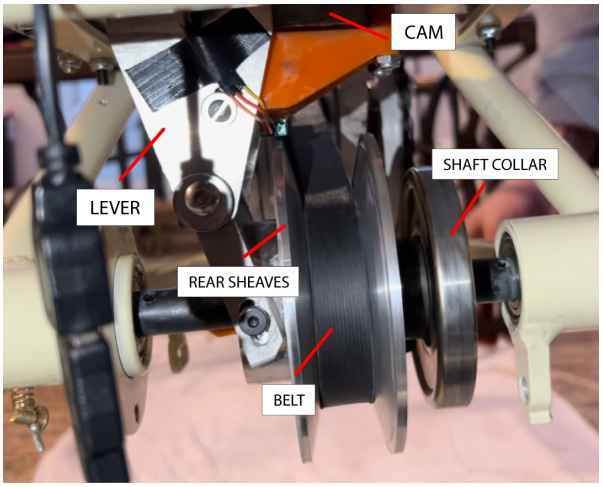
High Level Strategy

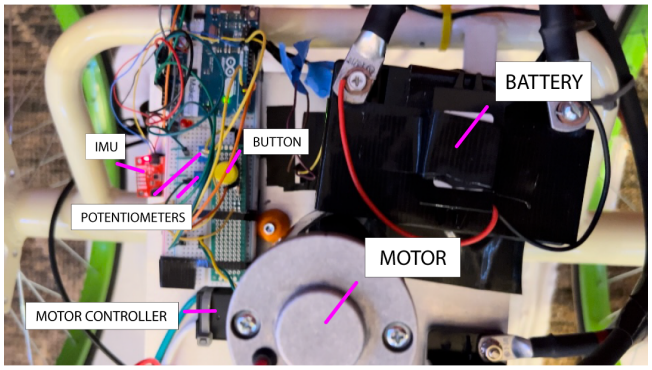
Our team decided on a continuously variable transmission (CVT) in order to infinitely vary the gear ratio of a bicycle and improve ride smoothness.
Our device achieved most of our specifications, however we did run into some minor issues such as belt slippage. This is due to us pivoting to lower force springs during integration as we were having difficulty assembling the sheave and belt compression system with the springs originally purchased. The trike is still capable of changing gear ratios while in motion, the main objective of the project. The IMU sensor was also integrated with the mechanical components, shifting to a lower gear ratio when positive pitch is detected. Overall, the project can be considered a success in the integration of mechanical and electrical components, even if some behaviors require further refinement.

Device Diagrams

The first photos display the CAD drawing, with the bicycle frame transparent so all parts can be seen. Given the difficult installation of our mechanism, close-ups are taken of the motor, sensors, and hardware.







Autobike Engineering Calculations		
Rider power and torque		
Rider weight	77	kg
Muscle multiplier	1	
Max pedal force	755.37	N
Crank arm length	0.175	m
Max input torque	132.18975	Nm
Max rider cadence	60	rpm
Max crank angular velocity	6.283185307	rad/s
Continuous power (empirical)	300	W
Peak power (calculated)	830.57	W

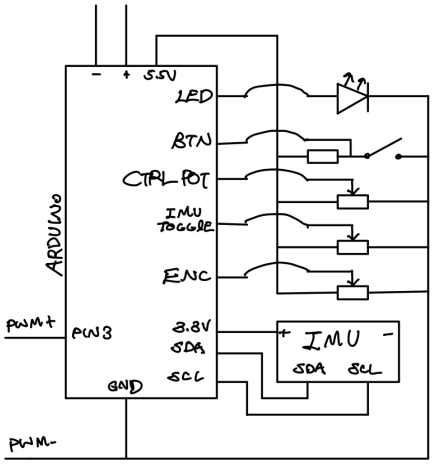
Transmission gear ratio specifications		
Range	Lower bound	Upper bound
First stage	1.81:1	NA
Second stage	0.577:1	1.73:1
Overall	1.04:1	3.13:1

Belt choice, specifications, and loads		
Max belt drive torque	73.05223026	Nm
Max rpm	108.5714286	rpm
Cross sectional area of belt	145.161	mm^2
Max belt tension	20.12998815	MPa
Max CVT Belt Stress	240	MPa

CAM sizing, lever sizing, motor spec		
Max torque	73.05223026	Nm
Min sheave rad	0.0254	m
Max tension	2876.072058	N
Parting force	1529.234639	N
Sheave distance	10	mm
Lever arm distar	30	mm
Cam min radius	25	mm
Cam max radius	40	mm
Cam "angle"	0.09520427991	rad
Lever ratio	1.5	_:1
Cam max latera	183.0964339	N
Cam max torque	7.323857355	Nm
	74.65705764	kg cm
CIM motor torqu	1.5	Nm
Min Planetary ra	4.88257157	

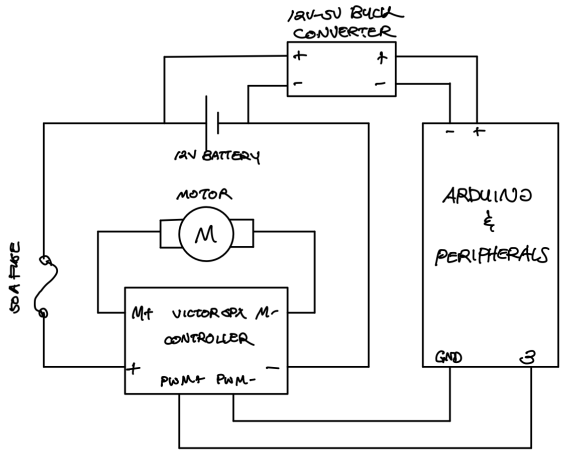
Critical Designs and Calculations

The CVT mechanism chosen for the project is the Reeves drive. This type of transmission utilizes two matched pairs of pulley sheaves that are actively driven on one side and spring loaded on the other. In order to size our components, we started by figuring out the pedal force a rider could produce and then determined the optimal chain drive ratio and CVT ratio range. We decided on an ideal range of 1.04:1 through to 3.13:1 (comparable to most road bikes). Based on this force and ratio, we then calculated the minimum belt size, sheave parting force, cam size, lever length, and minimum motor ratio. We opted for a planetary gearbox for its compact size and a Cam and lever to give us three total stages of mechanical advantage.

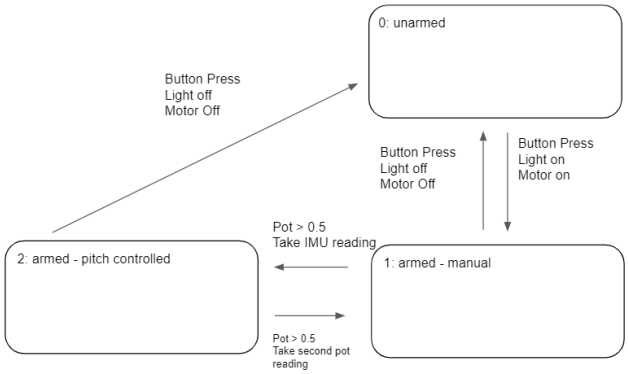


Circuit Diagram

The main elements of the circuit operating the motor are the 12V battery, Arduino, motor and the controller. The 12 V battery feeds into the Victor SPX micro- controller with a 50 A fuse connected in series on the positive lead for surge protection. The motor controller is fed a PWM signal from the Arduino based on sensor input and logic. The motor controller then feeds the appropriate voltage for motor operation. The Arduino determines the PWM signal from 4 inputs: a button serving as an arm/disarm toggle, a control potentiometer that directly controls the angular position of the cam, an encoder potentiometer that reads



the position, an IMU toggle potentiometer that determines how the position is controlled, and an IMU that gives accelerometer and gyroscope data.



State Transition Diagram

After setup is completed, the state machine begins in state 0 where all outputs are disconnected, and the motor remains inactive no matter the input. If the button is pressed in state 0, the Arduino arms the motor (state 1/2) and it is able to be controlled by inputs. If the IMU toggle potentiometer voltage is in the lower half it remains in state 1 where the CAM angular position is controlled by the voltage of the control potentiometer. If in the upper half, the control goes to IMU, where the pitch calculated from the accelerometer and gyroscope data controls the angular position. Pressing the button from either armed state will place the machine in state 0, turning the motor off.

Reflection

After this project, one key lesson our team learned was that it’s difficult to build something off of existing products. Integrating our system with the tricycle was the hardest part of our project - as many of our attachments needed modifications because they didn’t fit with our frame. In the future we also agreed more thought should be given to designing for assembly, as it was we had to lower our project goals because of the inability to compress the tensioning springs. In addition, machining custom parts takes much more time than expected - especially for metal parts. Given the time constraints of the project and capability of our members, we should have simplified the scope of our project. Everyone has different levels of skills and it’s efficient to employ them to the best of their abilities - while being able to recognize individual limits. Our members used this strategy to our advantage as we had individuals who focused on machining, electronics, CAD, and logistics. Strong communication allowed for us to keep on schedule and figuring out where we struggled during our project timeline.

Appendix

Bill of Materials

Autotrike's Purchase Portfolio								**Total (Projected):**	**$ 839.72**
Item Name	**Description**	**Purchase Justification**	**Serial Number / SKU**	**Price (ea.)**	**Quantity**	**Vendor**	**Link to Item**	**Notes**	**Subtotal**
Aluminum Disc	5" Diameter, 3" Length	used for the pulley sheaves	1610T48	$ 47.27	1	McMaster	https://www.mcmaster.com/catalog/128/4075		$ 47.27
Variable Speed V Belt	36.3 in circumference	used for pulley belt	1922V363	$ 17.49	1	V Belt Guys	https://www.vbeltguys.com/products/1922v363-variable-speed-belt?_pos=1&_sid=be460663b&_ss=r		$ 17.49
Hall effect sensor	Magnetic sensor	used for measuring bike speed	A3144	$ 5.69	1	MUZHI	https://www.amazon.com/Effect-Magnetic-Sensor-Arduino-MXRS/dp/B085KVV82D/ref=sr_1_4?crid=395XD6Y425TGX&keywords=hall+effect+sensor&qid=1665262668&qu=eyJxc2MiOiI0LjU1IiwicXNhIjoiMy43NCIsInFzcCI6IjMuNDYifQ%3D%3D&s		$ 5.69

							prefix=hall+effect+sensor%2Caps%2C126&sr=8-4		
Tricycle	Adult Tricycle	used for project integration	N/A	$ 240.00	1	Facebook MarketPlace		Zelle Receipt from Weston	$ 240.00
Motor and Gearbox	VEX CIM 12V motor with Planetary gear box	used to drive the CVT, encoder included	217-2000	$ 152.94	1	VEX	https://www.vexrobotics.com/versaplanetary.html		$ 152.94
Motor Controller	Victor SPX	motor controller for the motor	Victor 883	$ 50.00	1	VEX	https://www.vexrobotics.com/217-9191.html		$ 50.00
Battery	Lead Acid 12V Battery	used to provide power to necessary components	FAYTX5L	$ 59.99	1	Costco	https://www.costco.com/interstate-batteries-powersport-agm-battery-faytx5l.product.100655305.html		$ 59.99
Buck converter	12V to 5V DC converter	used to provide power to arduino	N/A	$ 9.59	1	Amazon	https://www.amazon.com/Converter-Module-Output-Adapter-Regulator/dp/B08RBWX2GL/ref=sr_1_3?keywords=12v+to+5v+usb&qid=1666318119&qu=eyJxc2MiOiI0LjUyIiwicXNhIjoiNC4yNCIsInFzcCI6IjQuMDcifQ%3D%3D&sprefix=12v+to+5v%2Caps%2C149&sr=8-3		$ 9.59

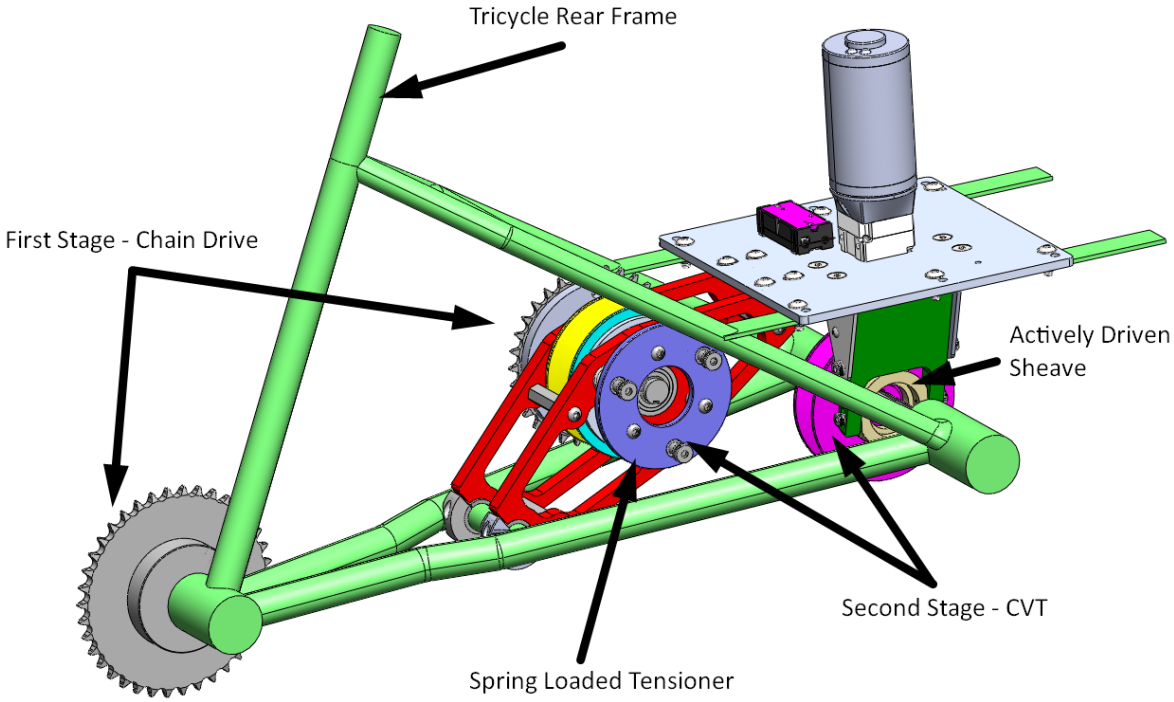
Battery Wires	low gauge wire to connect to the battery	need high amperage wire for motor	N/A	$ 13.87	1	Amazon	https://www.amazon.com/Gauge-Copper-Battery-Inverter-Cables/dp/B00SFXVMYW/ref=sr_1_4?crid=1330TJ752RC6&keywords=car+battery+cable&qid=1666324899&qu=eyJxc2MiOiI1Ljc2IiwicXNhIjoiNS41MSIsInFzcCI6IjUuMDkifQ%3D%3D&sprefix=car+battery+cabl%2Caps%2C142&sr=8-4		$ 13.87
Roller Chain Sprocket	ANSI 40 Chain, 38 Teeth, 3/4" Shaft	used for the input ratio for CVT drive	6236K196	$ 80.94	1	McMaster	https://www.mcmaster.com/6236K196/		$ 80.94
Roller Chain Sprocket	ANSI 40 Chain, 21 Teeth, 3/4" Shaft	used for the input ratio for CVT drive	6280K822	$ 38.46	1	McMaster	https://www.mcmaster.com/6280K822/		$ 38.46
Ring Shim	316 SS, 0.025", 3/4" ID	used as part of the CVT transmission system	97022A752	$ 6.86	1	McMaster	https://www.mcmaster.com/97022A752/		$ 6.86
Ball Bearing	Shielded, R12-2Z, 3/4" Shaft	used as part of the CVT transmission system	60355K603	$ 11.04	2	McMaster	https://www.mcmaster.com/60355k603/		$ 22.08
Clamping Two Piece Shaft	20mm Shaft, 1215 Carbon	used as part of the CVT	6063K19	$ 12.95	2	McMaster	https://www.mcmaster.com/6063k19/		$ 25.90

Collar	Steel	transmission system							
Oil Embedded Thrust Bronze Bearing	30mm Shaft, 60mm OD, 5mm thick	used as part of the CVT transmission system	2011N133	$ 11.61	1	McMaster	https://www.mcmaster.com/2011N133/		$ 11.61
Compression Spring	1.75" L, 0.75" OD, 0.5" ID, 221 lb/in spring rate	used as part of the CVT driving assembly	9657K96	$ 20.55	2	McMaster	https://www.mcmaster.com/9657K96/		$ 41.10
Shoulder Screw	Steel, 1/2" Shoulder, 1" Length, 3/8"-16 Thread	used as part of the CVT driving assembly	91259A712	$ 3.27	2	McMaster	https://www.mcmaster.com/91259a712/		$ 6.54
Spacers	Zinc-Plated Steel Unthreaded Spacer, 1/2" OD, 1/4" Long, for 1/4" Screw Size	used as part of the CVT driving assembly	92415A862	$ 2.52	2	McMaster	https://www.mcmaster.com/92415A862/		$ 5.04
Spacers	Aluminum Unthreaded Spacer, 13 mm OD, 13 mm Long, for M6 Screw Size	used as part of the CVT driving assembly	94669A178	$ 1.09	2	McMaster	https://www.mcmaster.com/94669A178/		$ 2.18
Hex Machine Screws	Button Head Hex Drive Screw, Black-Oxide	used as part of the CVT driving assembly	91239A328	$ 12.82	1	McMaster	https://www.mcmaster.com/91239A328/		$ 12.82

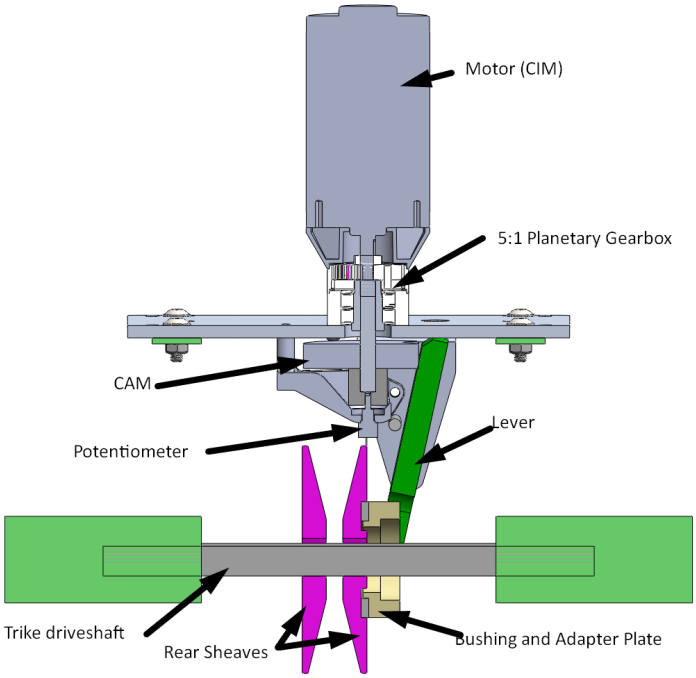
	Alloy Steel, M6 x 1.00 mm Thread, 30 mm Long								
Keyed Shaft	1045 Carbon Steel Keyed Rotary Shaft, Fully Keyed, 3/4" Diameter, 6" Long	used as part of the CVT driving assembly	1497K115	$ 15.45	1	McMaster	https://www.mcmaster.com/1497K115/		$ 15.45
Retaining Ring	3/4" OD, Black-Phosphate 1060-1090 Spring Steel	used as part of the CVT driving assembly	97633A250	$ 15.38	1	McMaster	https://www.mcmaster.com/97633A250/		$ 15.38
Thrust Bearing	Ultra-Low-Friction Oil-Embedded Thrust Bearing for 40 mm Shaft Diameter, 60 mm OD, 4 mm Thick	used as part of the CVT driving assembly	7421K65	$ 21.06	1	McMaster	https://www.mcmaster.com/7421K65/		$ 21.06
Motor Encoder	VersaPlanetary Integrated Encoder	used as part of the CVT driving assembly	217-5046	$ 49.99	1	VEX	https://www.vexrobotics.com/versaplanetary.html		$ 49.99
Aluminum Bar Stock	6061 Aluminum 1/2" Thick 5" Wider 3'	used as machining stock for CVT	8975K217	$ 104.42	1	McMaster	https://www.mcmaster.com/8975K217/		$ 104.42

	Long	assembly							
Aluminum Sheet Stock	Multipurpose 6061 Aluminum, 1/4" Thick x 12" Wide, 1 Foot Long	used as machining stock for CVT assembly	9246K13	$ 35.61	2	McMaster	https://www.mcmaster.com/9246K13/		$ 71.22
11/17 Mcmaster Order	Main Hardware Order	Screws, springs, nuts, etc.		$ 262.60	1	McMaster			$ 262.60
11/18 Mcmaster Order	Shoulder Bolts			$ 26.59	1	McMaster			$ 26.59
11/10 VexPro Order	Motor, motor controller, gearbox	For active sheave system		$ 199.09	1	Vexpro			$ 199.09

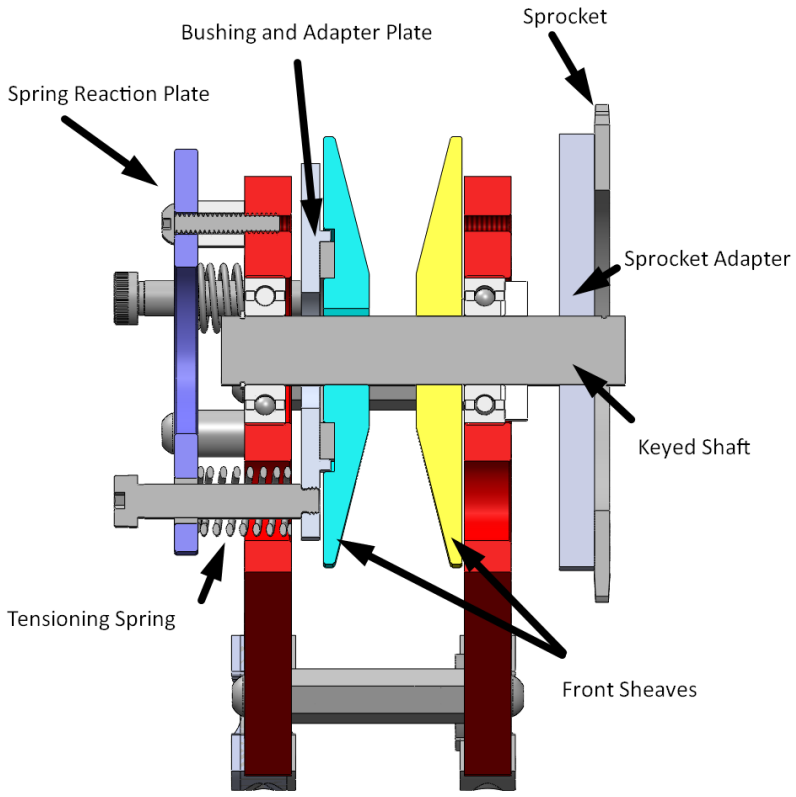
CAD



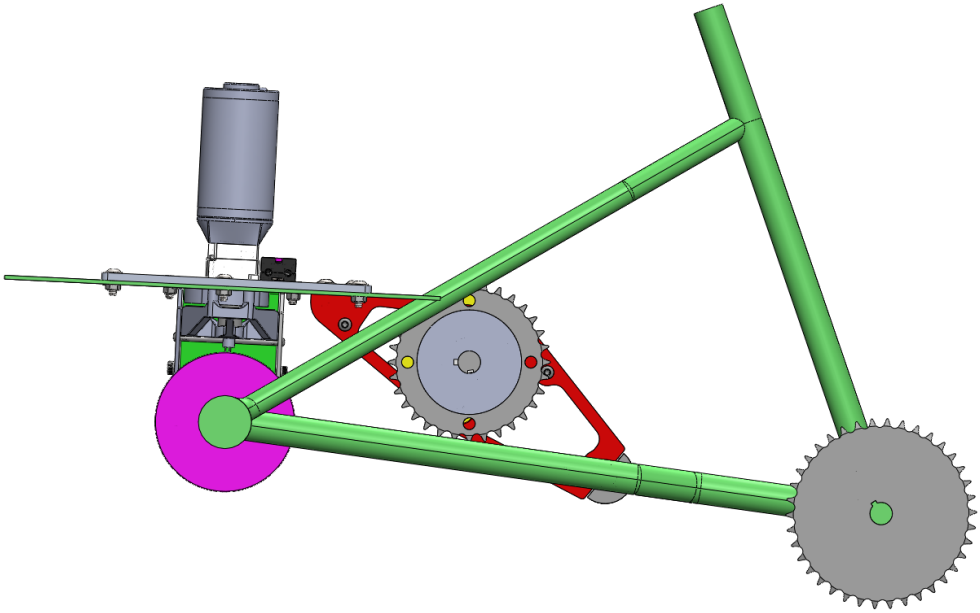
Top Level Assembly



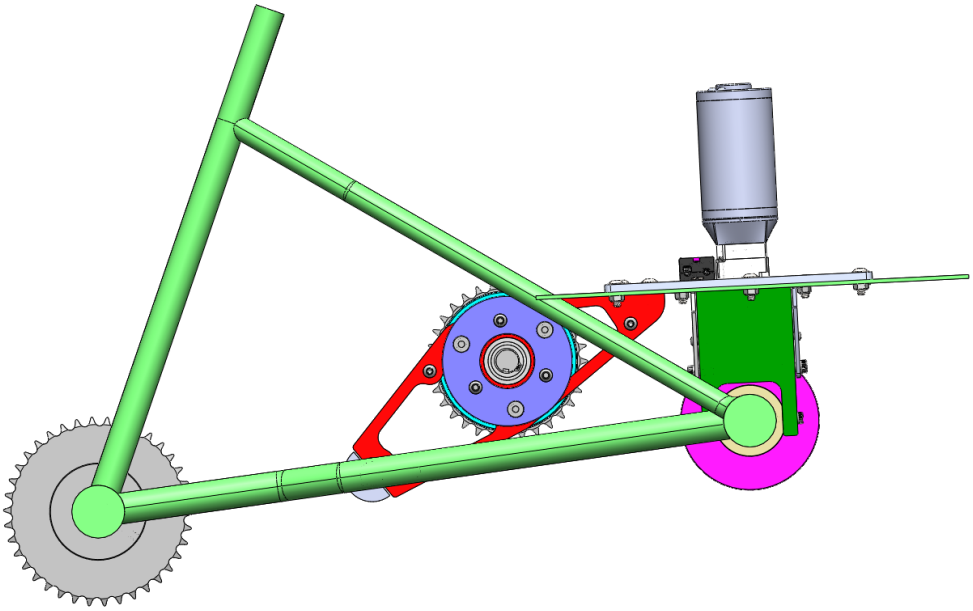
Rear Stage Front View Cross Section



Front Stage Rear View Cross Section (Frame Hidden)



Right Side View



Left Side View

Event Driven Programming

cvt_bike.ino

```
#include <Arduino.h>
#include <math.h>
#include <Wire.h>
#include <Servo.h>

#define LED 13
#define POT A0
#define ENC A1
#define BTN 2
#define IMUPOT A2
#define PWM0 3

//Setup variables -----------------------------
volatile bool buttonIsPressed = false;
Servo victor_spx;
int state = 0;
int camAngle = 180;
int rotMult = 1;
int mapReading;
int currTheta;

void btn_isr() {  // the function to be called when button interrupt is
triggered
  buttonIsPressed = true;
  if (state != 0) {
    digitalWrite(LED, LOW);
    victor_spx.detach();
    Serial.println("State 0: Unarmed");
    state = 0;
  } else {
    digitalWrite(LED, HIGH);
    victor_spx.attach(PWM0);
    Serial.println("State 1: POT Control");
    state = 1;
  }
}

bool checkButton() { //Check if the button has been pressed, reset to check
var to false
```

```
  if (buttonIsPressed){
    buttonIsPressed = false;
    return true;
  } else {
    return false;
  }
}

bool checkIMU() { //Check if the IMU toggle potentiometer is in the lower
half, return true if the IMU should be in control
  int reading = analogRead(IMUPOT);
  if (reading > 511) {
    return true;
  } else {
    return false;
  }
}

int potRead() { //Map control potentiometer to a corresponding CAM angle
  int rawReading = analogRead(POT);
  mapReading = 0.6*rawReading + 0.4*mapReading;
  int degree = 10*map(mapReading, 0, 1023, 9, 30);
  return degree;
}

void moveServo(int desAngle) {
  int deltaTheta = desAngle - map(analogRead(ENC), 0, 1023, 315, 0);
  while (abs(deltaTheta) > 5) {
    if (state == 1) {
      desAngle = potRead(); //refresh input incase potentiometer is changed
while loop is running (troubleshooting failsafe)
    }
    deltaTheta = desAngle - map(analogRead(ENC), 0, 1023, 315, 0);
    int input = 0.1*deltaTheta;
    victor_spx.write(94+input); //Inherent pressure on the cam causes bias
causing stop value to be 94 instead of 90
    printStatus();
  }
  victor_spx.write(94);
}

void printStatus() { //Print status function for troubleshooting
    Serial.print(state);
```

```
    Serial.print(", ");
    Serial.print(potRead());
    Serial.print(", ");
    Serial.print(getTheta());
    Serial.print(", ");
    Serial.println(map(analogRead(ENC), 0, 1023, 315, 0));
}
void setup() {
  Serial.begin(9600);
  pinMode(BTN, INPUT);
  pinMode(LED, OUTPUT);
  attachInterrupt(digitalPinToInterrupt(BTN), btn_isr, RISING);
  imuSetup();
  Serial.println("Setup Complete");
  Serial.println("State 0: Unarmed");
}

void loop() {
  switch (state) {
    case 0: //UNARMED
      printStatus();
      break;

    case 1: //ARMED
      if (checkIMU()) {
        Serial.println("State 2: IMU Control");
        state = 2;
      }
      moveServo(potRead());
      printStatus();
      break;

    case 2:
      if (checkIMU() == false) {
        Serial.println("State 1: POT Control");
        state = 1;
      }
      currTheta = 0.6*currTheta + 0.4*getTheta();
      if (abs(currTheta) > 30) {
        currTheta = 30*currTheta/abs(currTheta);
      }
      moveServo(map(currTheta, -30, 30, 80, 300));
      printStatus();
```

```
      break;
    }
}
```

Imu_helper.ino

```
#include <Wire.h>
#include <math.h>

#define IMU 107
bool thetaNeg = false;
int theta;
float pitchratesum;

float getzxl() {
  int upperbyte; int lowerbyte;
  int reading;
  float zaccel;
  byte error; //variable for error and I2C address

  Wire.beginTransmission(IMU);
  Wire.write(byte(0x2C)); //zxl address
  error = Wire.endTransmission();
  if (error == 4) {
    Serial.println("Error when setting ZXL register pointer");
  } else if (error != 0) {
    Serial.println("Address not found when looking for ZXL register");
  }

  Wire.requestFrom(IMU, 2); //get 2 bytes from zxl address
  if (2 <= Wire.available()) {
    lowerbyte = Wire.read();
    upperbyte = Wire.read() << 8;
    reading = upperbyte | lowerbyte;
  }

  if (reading > 32767) {
    reading = reading - 65536;
  }
  zaccel = float(reading)/16560;
  if (zaccel > 1.0) {
    zaccel = 1.0;
```

```
  }
  return zaccel;
}

int getpitchw() {
  int upperbyte; int lowerbyte;
  int reading;
  byte error; //variable for error and I2C address

  Wire.beginTransmission(IMU);
  Wire.write(byte(0x24)); //pitch rate address
  error = Wire.endTransmission();
  if (error == 4) {
    Serial.println("Error when setting pitch rate register pointer");
  } else if (error != 0) {
    Serial.println("Address not found when looking for pitch rate
register");
  }

  Wire.requestFrom(IMU, 2); //get 2 bytes from zxl address
  if (2 <= Wire.available()) {
    lowerbyte = Wire.read();
    upperbyte = Wire.read() << 8;
    reading = upperbyte | lowerbyte;
  }

  if (reading > 32767) {
    reading = reading - 65536; //65536 nominal
  }
  return reading - 185;
}

void imuSetup()
{
  Wire.begin(); // Wire communication begin
  Serial.begin(9600); // The baudrate of Serial monitor is set in 9600
  while (!Serial); // Waiting for Serial Monitor

  //SET CTRL_REG1_G
  Wire.beginTransmission(IMU);
  Wire.write(byte(0x10));
  Wire.write(byte(0xA0));
  Wire.endTransmission();
```

```
  //SET CTRL_REG2_G
  Wire.beginTransmission(IMU);
  Wire.write(byte(0x11));
  Wire.write(byte(0xA0));
  Wire.endTransmission();

  delay(500);
}

int getTheta() {
  float zxl = getzxl();
  float pitchw = getpitchw();
  pitchratesum += pitchw;
  if (zxl >= 0.9999999f) {
    theta = 0;
    pitchratesum = 0;
  } else {
    if (pitchratesum < 0) {
      theta = 0.5*(-acos(zxl)*360/6.28) + 0.5*theta;
    } else {
      theta = 0.5*(acos(zxl)*360/6.28) + 0.5*theta;
    }
  }
  return theta;
}
```

Custom Part Tracker

Priority	Qty	COTS Parts	Thickness (in)	Process	Post WJ ops	Drawing Done	Material	Status	Owner
A	1	Sheave Driven	0.5	Waterjet, Lathe	lathe	Y	Al 6061	Done	Max
A	1	Sheave Driven Level	0.5	Waterjet, Lathe	lathe	Y	Al 6061	Done	Max
A	1	Sheave Driving	0.5	Waterjet, Lathe	lathe	Y	Al 6061	Done	Max
A	1	Sheave Driving Spring	0.5	Waterjet, Lathe	lathe	Y	Al 6061	Done	Max
C	2	Mounting Plates	0.5	Waterjet, Mill	bearing bore, tap	Y	Al 6061	Done	Max
B	1	Spring Load Plate	0.5	Waterjet, Lathe, Tap	lathe, tap	Y	Al 6061	Done	Max
B	1	Lever Arm	0.5	Waterjet, Mill	mill	Y	Al 6061	Done	Max
B	1	Lever Pad	0.5	Waterjet, Lathe, Mill	lathe, mill, tap	Y	Al 6061	Done	Max
B	1	Cam	0.5	Waterjet	none	N/A	Al 6061	Done	Max
A	1	Prototype Plate	0.5	Waterjet	bearing bore	N/A	Al 6061	Done	Max
B	1	Top Plate	0.25	Waterjet	countersink, new holes	Y	Al 6061	Done	Max
C	1	Battery Cage	0.25	Waterjet	none	N/A	Al 6061	Done	Max
C	1	Spring Reaction plate	0.25	Waterjet	bushing bore	Y	Al 6061	Done	Max
B	2	Lever Mount	0.125	Waterjet	drill and bend	Y	Al 3003	Done	Max
C	1	Intermediate Shaft	0.75"	Lathe		Y	1497K115	Done	Max
C	3	Mount Plate Standoff (post machine)		Bandsaw		Y see notes	95947A562	Done	Max
C	1	Lever Plate Standoff (post machine)		Bandsaw		Y see notes	93330A553	Done	Max
C	2	Sheave Spacer		3D Print		N/A	Nylon preferred, PLA okay?	Done	Max
C	2	Sprocket Spacer		3D Print		N/A	Nylon preferred, PLA okay?	Done	Max
C	1	Sprocket adapter		Mill		N	6061 stock from dennis	Done	Max