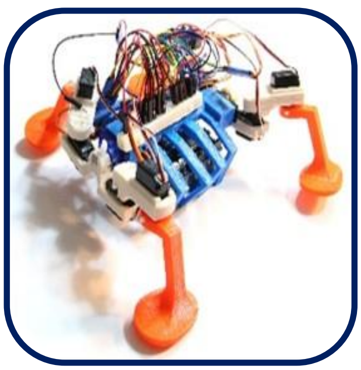
OLD DOMINION UNIVERSITY
Frank Batten College of Engineering & Technology

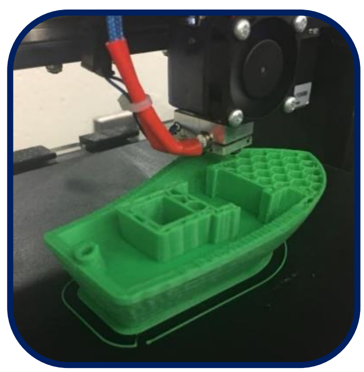
125 / 116 Kaufman Hall
Old Dominion University
Norfolk, VA 23529

VETERAN'S MAKER WORKSHOP
FEATURING BIOINSPIRED ROBOTICS

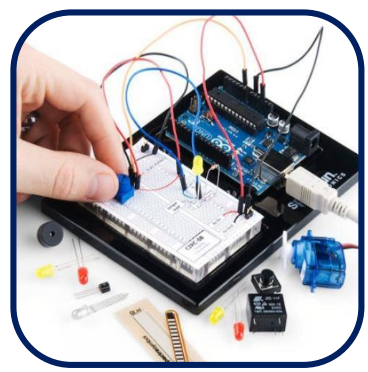


Robotic Design Principles
Dr. Krishna Kaipa

3D Design
Dr. Vukica Jovanovic



3D Printing
Dr. Karina Arcaute



Programming of Microcontrollers
Dr. Otilia Popescu

EAGER: Understanding the Impact of Making on Veterans in Pursuing STEM Degrees
Project #1749566, Funded by the National Science Foundation

POC: Dr. Tony Dean | adean@odu.edu | 757-683-7121

Monarch Maker Workshop Schedule

DAY 1	Activities	Duration	
8:00 am – 9:00 am	***Pre-workshop Assessment Surveys***	1 hr.	A. Dean
9:00 am – 9:45 am	***Bio-Inspired Robotics:*** Introduction to principles of bio-inspired robotics, legged robots, and walking gaits.	45 min.	K. Kaipa
9:45 am – 10:00 am	**Break**	15 min.	
10:00 am – 12:00 pm	***Arduino:*** Introduction to microcontrollers, Arduino, and programming. Hands-on activity with LEDs, single motors, and multiple motors	2 hrs.	O. Popescu
DAY 2	**Activities**	Duration	
8:00 am – 9: 00 am	***Making:*** Intro to 3D Printing and Additive Manufacturing Technologies.	1 hr.	K. Arcaute
9:00 am – 10:00 am	***Computer Aided Design (CAD):*** Introduction to CAD. Keychain Activity	1 hr.	V. Jovanovic
10:00 am – 10:15 am	**Break**	15 min.	
10:15 am – 11:05 am	***Hands-On Making:*** Slicing **-** creating G code from STL designs and preparing them to be 3D printed.	50 min.	K. Arcaute
11:05 am – 12:00 pm	**Design:** Parametric Modeling Fundamentals	55 min.	V. Jovanovic
DAY 3		Duration	
8:00 am – 10:00 am	***Assembly:*** Assembly of bio-inspired robots with pre-printed parts and servo motors	2 hrs.	K. Kaipa K. Arcaute
10:00 am – 10:15 am	**Break**	15 min.	
10:15 am – 12:00 pm	***Electrical wiring:*** Electrical wiring between servo motors, batteries, and switch	1 hr. 45 min.	K. Kaipa O. Popescu
DAY 4		Duration	
8:00 am – 9:45 am	***Programming:*** Programming code for: a) testing each leg and b) walking gait	1 hr. 45 min.	K. Kaipa
9:45 am – 10:00 am	***Break***	15 min.	
10:00 am – 10:45 am	***Testing:*** Testing of robot walking and readjustment by analyzing possible failures and making improvements.	45 min.	K. Kaipa V. Jovanovic
10:45 am – 11:15 am	***Race to Finish***	30 min	K. Kaipa
11:15 am – 12:00 pm	***Workshop Assessment Survey***	45 min.	A. Dean

Module

Biologically Inspired Robotics

Introduction to Principles of Bio-inspired Robotics

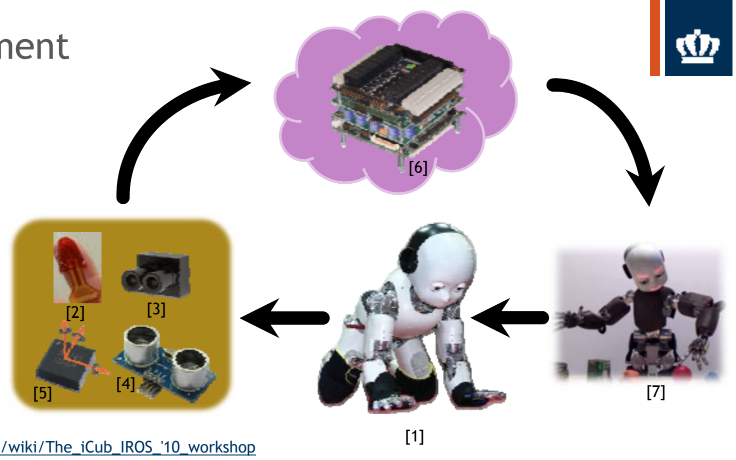
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Outline

- Introduction to Traditional Robotics
- Biological Inspiration
- Legged Robotic Locomotion
- Bio-inspired Gaits
- Leg Mechanisms and Kinematics
- Examples of 3D-printed Bio-inspired Robots
- Project Goal

What is a Robot?

- Embodiment
- Sense
- Think
- Act



[1] http://wiki.icub.org/wiki/The_iCub_IROS_'10_workshop
[2] http://www.calit2.uci.edu/calit2-newsroom/itemdetail.aspx?cguid=776d6aa6-0a83-458c-a770-c034d041fa50
[3] http://www.vision-systems.com/articles/2012/10/epix-stereo-camera-captures-8-to-12-bit-images-at-up-to-340-frames-per-second.html
[4] http://letsmakerobots.com/content/hc-sr04-ultrasonic-sensor
[5] http://www.designworldonline.com/6dof-sensors-improve-motion-sensing-applications/
[6] http://www.signal11.us/io.html
[7] https://www.youtube.com/watch?v=ZcTwO2dpX8A

Traditional Robotic Locomotion

- Engineered solutions to locomotion
  - Dependence on wheels, rotors, airfoil wings for basic propulsion
- Examples
  - Unmanned ground vehicles (Robotic self-driving cars, tracked robots, etc)
  - Unmanned underwater/surface vehicles (Robotic boats/submarines)
  - Unmanned aerial vehicles (Robotic helicopters, quadcopters, etc)

http://www.clearpathrobotics.com/husky/

http://www.irobot.com/us/learn/defense/sugv.aspx

Biological Inspiration

- Animals/insects evolved distinct traits to perform reliably in unstructured environments
- Features like softness, compliance, and configurability to reduce complexity of interaction with surroundings

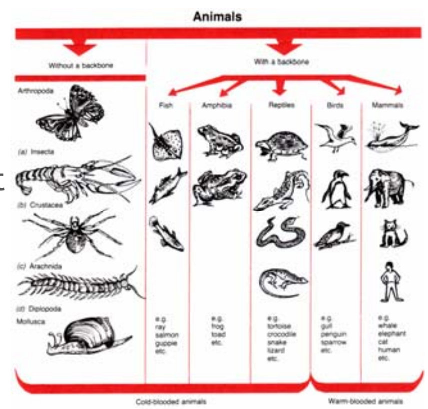
Bio-inspired insights can be used to synthesize locomotion and manipulation capabilities in robotic systems

Examples of Robotic Locomotion

(a, b) Examples of less biologically inspired solutions to locomotion – wheeled and tracked robots. (c,d) Examples of more bio-inspired solutions to locomotion – legged robots (quadrupedal and bipedal humanoid).

Source of Bio-inspiration

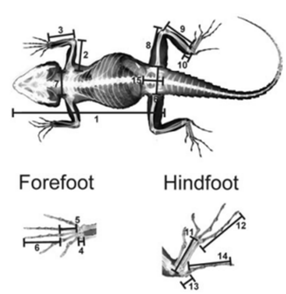
- Entire animal kingdom
  - Vertebrates
  - Invertebrates
- Examples
  - Fish locomotion ➔ Marine robot
  - Human hand ➔ Robotic gripper
- Principal traits
  - Morphology
  - Biomechanics
  - Control



http://mapsandmemories.edublogs.org/files/2012/01/classification-animals-qybz56.gif

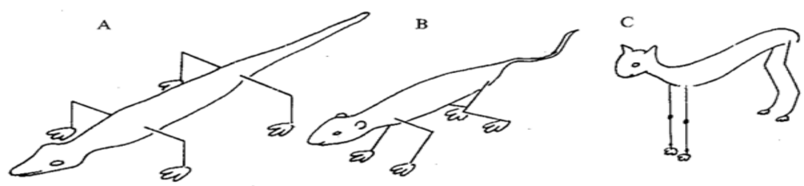
Morphology

- Refers to the overall body plan of the natural creature chosen as the inspiration source
  - Kinematic structure of the body
  - Geometrical shapes and aspect ratios of dimensions of various body segments
  - Compliance/stiffness properties of different body segments
  - Limbs specialized for propulsion versus those specialized for manipulation
- Morphology determines whether the animal can operate in a particular environment or not
- Morphology impacts how well the animal performs in a chosen environment

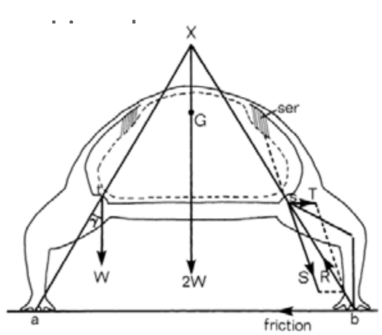


Animal's Body Plan Decides its Locomotion Mode

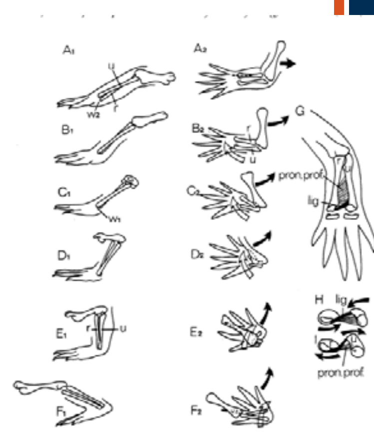
- Body structures of most animals are best suited to only one locomotion mode.
  - Purely aquatic creatures (e.g., fish, whales, dolphins, etc) are suitable for swimming alone
  - Terrestrial animals (e.g., quadrupeds like cheetah, horse, etc.,) are best suited for walking and running on land
- Body plans of reptilian family (e.g., lizards) and cursorial mammals (e.g., Cheetah) distinctly differ from each other, which reflects in their differing locomotion characteristics



Biomechanics



Statics of sprawling posture

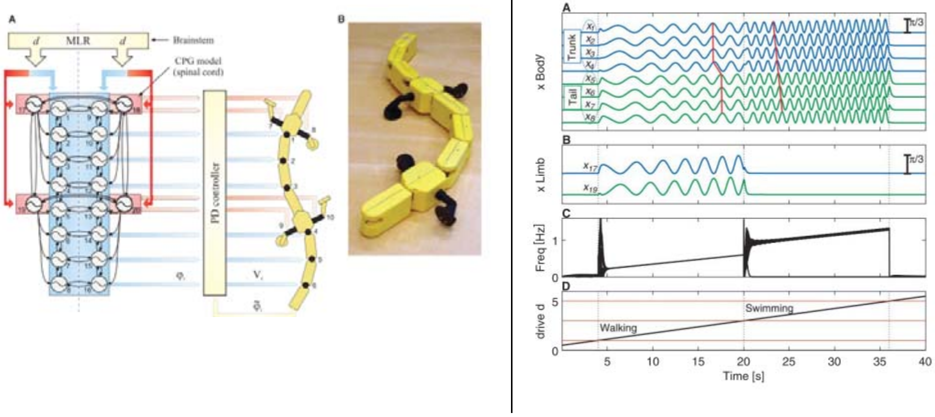


Integration of movements in shoulder, elbow, and wrist to maintain ground contact during stance phase

[Russell and Bels 2001]

Biological Control

- Central pattern generators (CPGs)



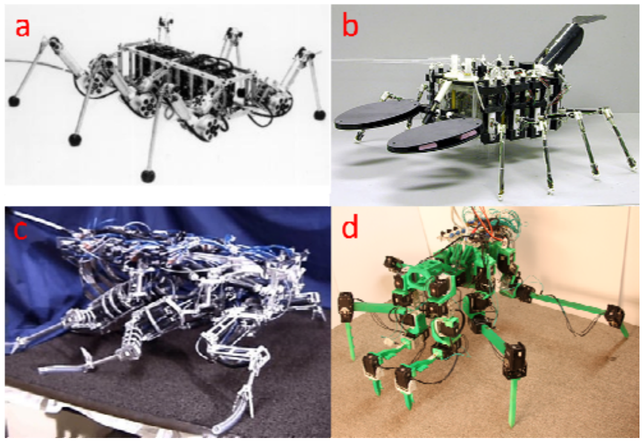
[Ijspeert et al. 2007]

Examples of Bio-inspiration

- Gorilla http://www.youtube.com/watch?v=2Npc5QlS6Iw
- Turtle http://www.youtube.com/watch?v=5e-VkSYA1NU
- Lizard http://www.youtube.com/watch?v=7-5tyRvMGvY
- Bear http://www.youtube.com/watch?v=PfnX3i5kwUU
- Sloth http://www.youtube.com/watch?v=eotEEUNatKY
- Rabbit/Ferret http://www.youtube.com/watch?v=kxDymYeKV3Q
- Komodo Dragon http://www.youtube.com/watch?v=n6Riq-d4W_o
- Gecko http://www.youtube.com/watch?v=iv7VoFcYdnw

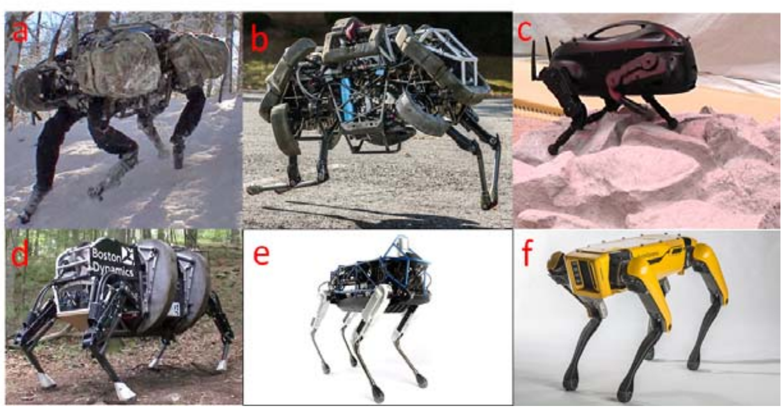
NSF Grant # 1749566
https://sites.wp.odu.edu/oduvetmaker/

Six-legged and Eight-legged Robots



Multi-legged robots that borrow inspiration from insects in which the legs are spread out and move sideways: a) Stick-insect inspired hexapedal robot b) Lobster-inspired eight-legged robot c) Ajax, a cockroach-inspired hexapedal robot, d) Mantis

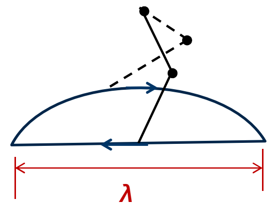
Four-legged Robots



Lineup of quadrupedal robots from Boston Dynamics: a) Big Dog, b) WildCat, c) Little Dog, d) LS3 e) Spot, and f) Spot Mini. [Source: https://www.bostondynamics.com/]

Locomotion Gaits

- Stride
  - A complete cycle of leg movements starting from the setting down of a foot to the next setting down of the same foot
- Stride frequency f
  - Number of strides taken in unit time
- Stride length λ
  - Distance travelled in one stride

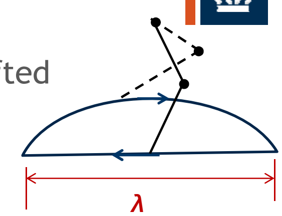


➔ Mean speed $u = \lambda f$

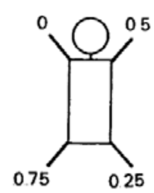
- Stance phase
  - Duration of the stride when the foot is in contact with the ground

Locomotion Gaits (Contd.)

- Swing phase
  - Duration of the stride when the foot is lifted and moved forwards



- Duty factor β
  - Fraction of the duration of the stride when the foot is in contact of the ground
- Relative phase
  - Stage of the stride when a foot is set down, expressed as a fraction of duration of the stride following the setting down of a arbitrarily chosen reference foot



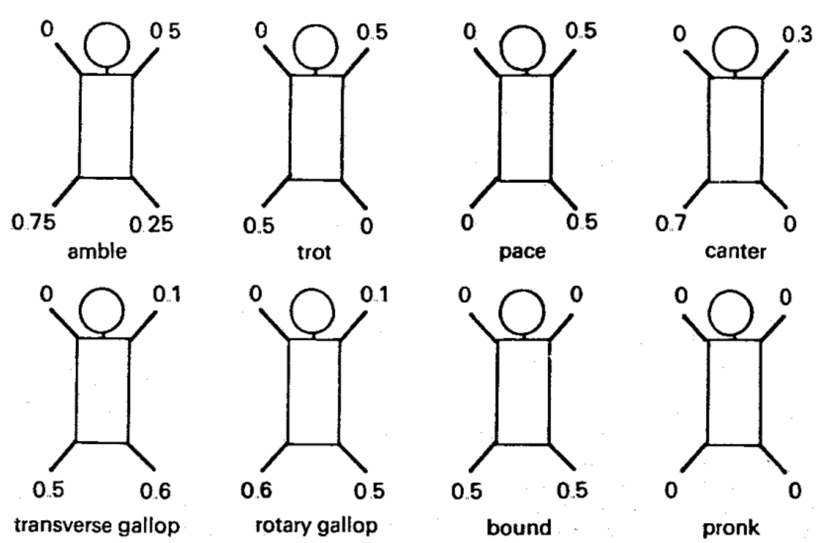
NSF Grant # 1749566
https://sites.wp.odu.edu/oduvetmaker/

Gait Classification

- Classification 1 (based on speed)
  - Walking: **ß > 0.5** (must be stages when both feet of a pair are on ground simultaneously)
  - Running: **ß < 0.5** (must be stages when both feet of a pair are off the ground simultaneously)

- Classification 2
  - Symmetrical: left and right feet of each pair have equal ß and relative phase differing by 0.5 (eg., walking and slow running)
  - Asymmetrical: left and right feet of each pair move in unison (eg., fast running and hopping)

Quadrupedal Gaits: Relative Feet Phases

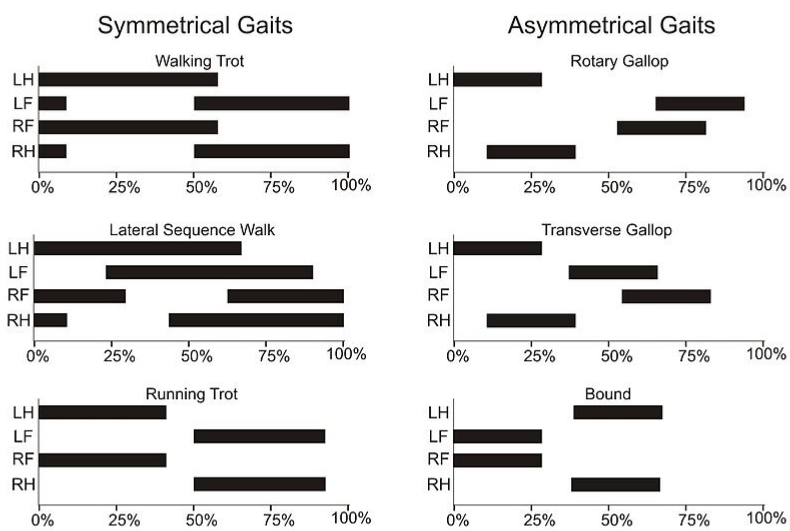


NSF Grant # 1749566

Gait videos

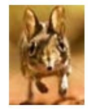
- Walk (four beat)
  - http://en.wikipedia.org/wiki/File:Muybridge_horse_walking_animated.gif
- Trot (two beat)
  - http://en.wikipedia.org/wiki/File:Trot_animated.gif
- Pace (two beat)
  - http://en.wikipedia.org/wiki/File:Muybridge_horse_pacing_animated.gif
- Canter (three beat)
  - http://en.wikipedia.org/wiki/File:Gallop_animated.gif
- Gallop (four beat)
  - http://en.wikipedia.org/wiki/File:Muybridge_race_horse_animated.gif

Gait Graphs



Effect of Size on Gaits

- Difference in sizes to describe gaits
  - Elephant's weight ≈ 10^6 X Shrew's weight
  - Elephants' bone length ≈ 10^2 X Shrew's bone length



Shrew

Elephant

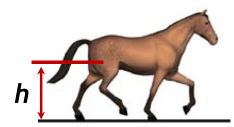
- For a given speed
  - Stride length proportional to size
  - Gait may be different
  - E.g., walking horse ≈ trotting dog ≈ galloping mouse
- Related animals have grossly different sizes but display a high degree of geometric similarity

Geometric Similarity

- For most mammals ranging from shrews to elephants
  - (Body + head) length α (body mass)$^{0.33}$
  - (Principal leg bone) length α (body mass)$^{0.35}$
- Geometric similarity ➔ Dynamic similarity

Dynamic similarity

- $Froude\ number = \frac{u^2}{gh}$
  - u is speed of travel
  - g is acceleration due to gravity
  - h is height of hip joint from ground



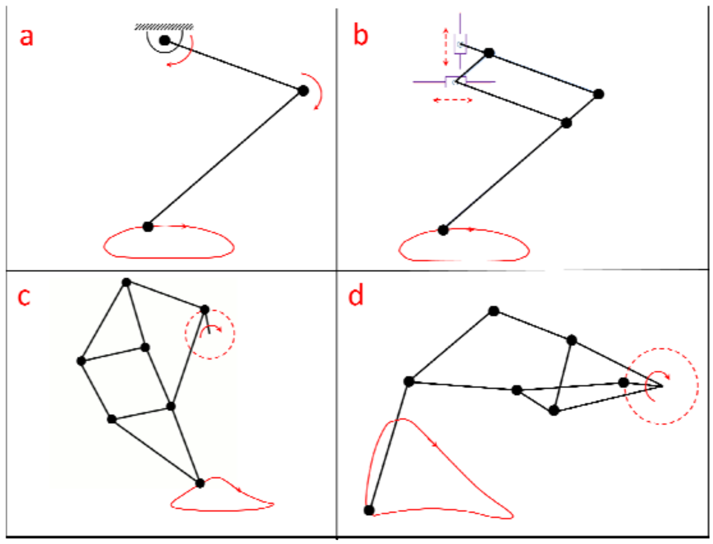
- Animals move in dynamically similar fashion when speeds are inversely proportional to square root of leg lengths

➔ dynamically similar gaits when Froude numbers are equal

Same Gait for Same Froude number

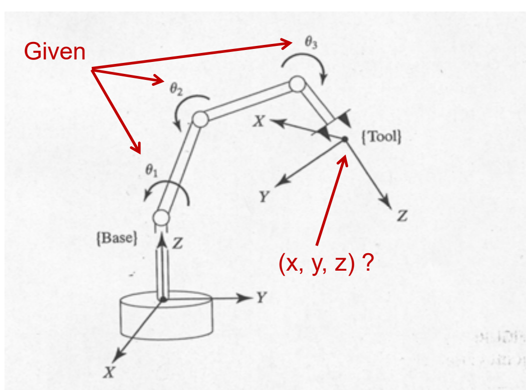
Froude No.	Mammal example	Height (m)	Speed (m/s)	Gait
0.1	Cat	0.22	0.5	Walking
	Camel	1.7	1.3	Walking
1.0	Cat		1.5	Symmetric running (Most species trot)
	Camel		4	Symmetric running (Pace)
2 - 3				Asymmetric running (Canter or gallop)

Leg Mechanisms

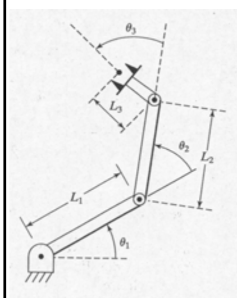


Forward Kinematics

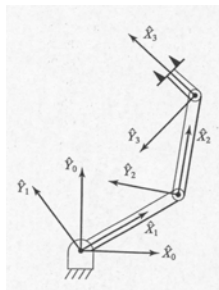
- Given joint parameters, determine the final end effecter location



Kinematics Example



Robot configuration



Frame assignment

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	L_1	0	θ_2
3	0	L_2	0	θ_3

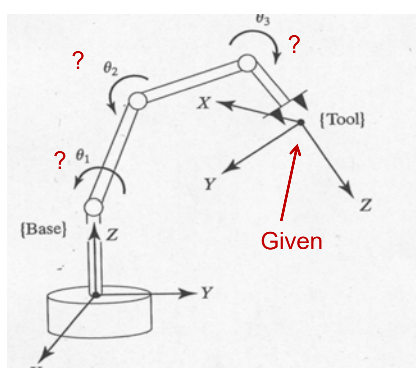
Denavit-Hartenberg parameters

$$
{}^{i-1}_{\ \ i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

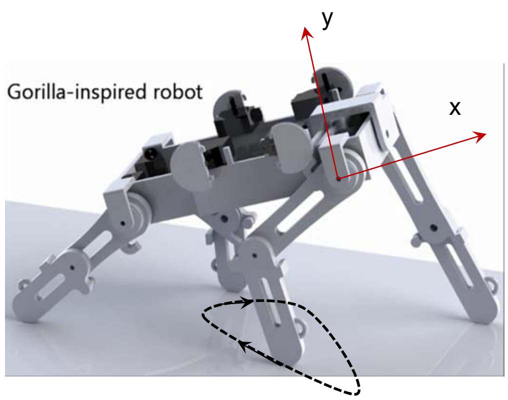
Transformation matrix

Inverse Kinematics

- Given desired end effecter position and orientation determine the joint parameters



Inverse Kinematics of a Walking Robot



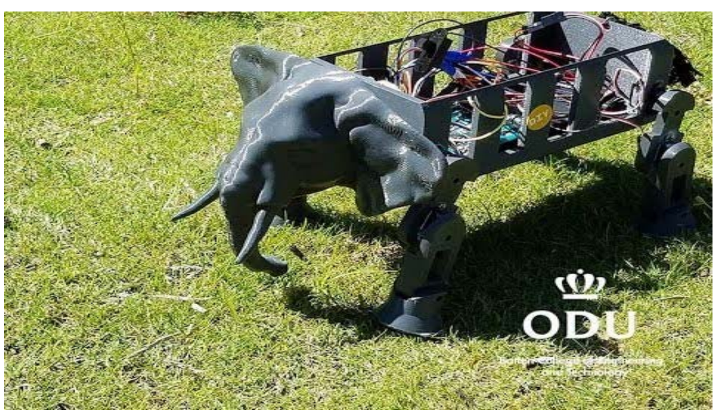
$(x^{(1)}, y^{(1)}) \rightarrow (\theta_1^{(1)}, \theta_2^{(1)})$

$(x^{(2)}, y^{(2)}) \rightarrow (\theta_1^{(2)}, \theta_2^{(2)})$

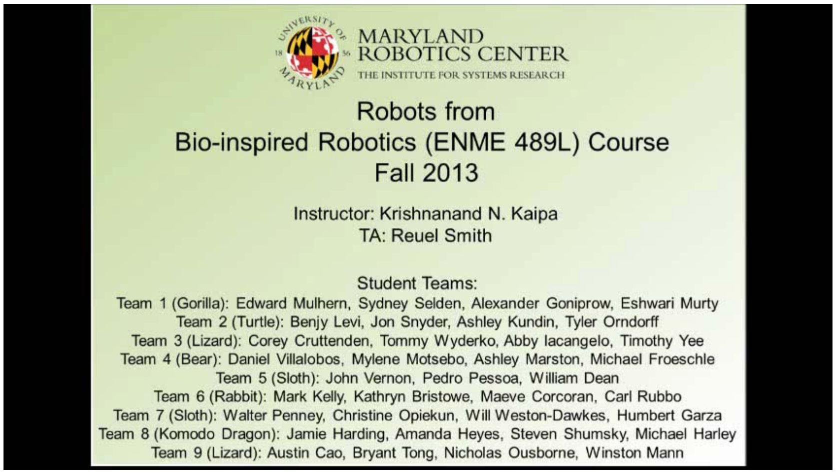
$\vdots$

$(x^{(n)}, y^{(n)}) \rightarrow (\theta_1^{(n)}, \theta_2^{(n)})$

Bio-inspired Robot Examples



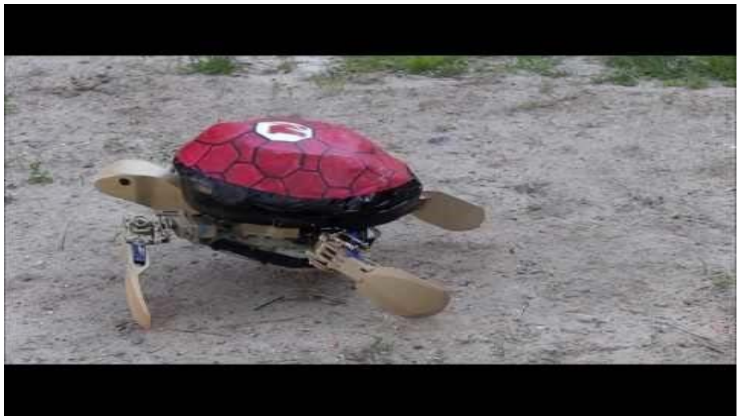
Bio-inspired Robot Examples (Contd.)



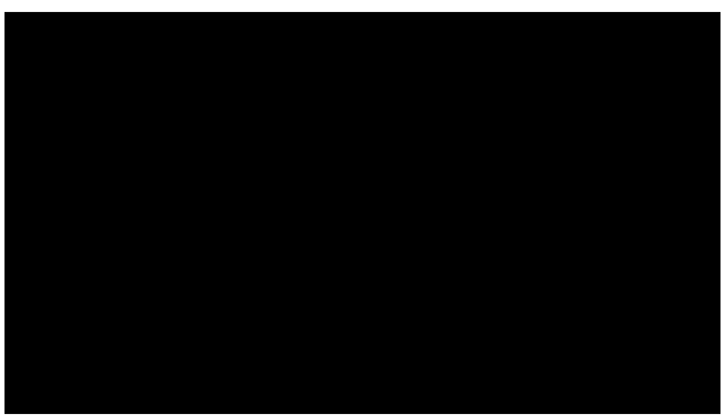
Turtle-inspired Amphibious Robot

A. Vogel, K. N. Kaipa, G. Krummel, H. A. Bruck, and S. K. Gupta. Design of a compliance assisted quadrupedal amphibious robot. *IEEE International Conference on Robotics and Automation* (ICRA 2014), Hong Kong, China, May 31-June 7, 2014.

Turtle-inspired Amphibious Robot



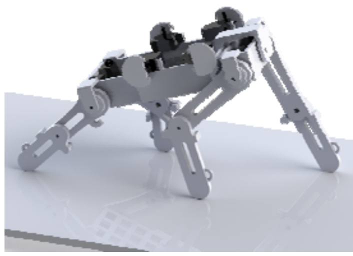
Horseshoe Crab inspired Self-righting Robot

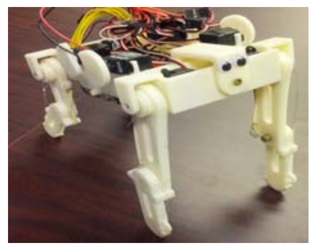


G. Krummel, K. N. Kaipa, and S. K. Gupta. Design of a horseshoe crab inspired amphibious robot for righting in surf zones. *ASME Mechanisms and Robotics Conference* (IDETC/CIE 2014), Buffalo, NY, August 17-20, 2014.

Project Goal

- Design a four legged bio-inspired robot that can perform the following function
  - Travel on a straight line 25 times its body's largest dimension in two minutes or less
- Demonstrate walking on two different surfaces
  - Parking lot
  - Carpeted floor





References

- J.J. Craig. *Introduction to Robotics: Mechanics and Control*. Prentice Hall; 3rd edition, 2003
- R. Alexander. The gaits of bipedal and quadrupedal animals, *The journal of Robotics Research:* 3(2), 1983, pp. 49—59.
- G. A. Bekey. Autonomous Robots: From Biological Inspiration to Implementation and Control. MIT Press, 2005.
- Karl Williams. Amphibionics: Build Your Own Biologically Inspired Reptilian Robot. McGraw-Hill/TAB Electronics, 2003.

Arduino Module

Introduction to Microcontrollers and Arduino Uno

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Objectives:

- Meet the Arduino microcontroller
- Hardware components
- Basics of electrical circuits
- Building a simple circuit on breadboard
- Programming with Arduino
- Controlling servo motors
- Controlling multiple motors

Acknowledgment: This work is supported by National Science Foundation grant 1749566

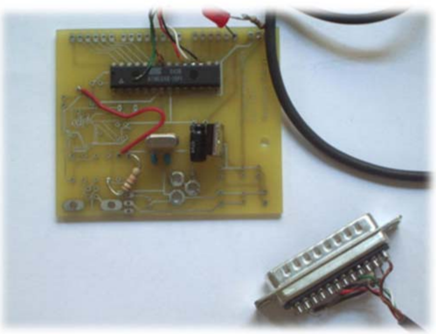
What is a Microcontroller?

- "A microcontroller is a very small computer that has digital electronic devices (peripherals) built into it that helps it control things. These peripherals allow it to sense the world around it and drive the actions of external devices."
  http://www.arduinoclassroom.com/index.php/arduino-101
- A small computer on a single chip, containing a processor, memory, and input/output pins.
- It is an "embedded computer system" that continuously repeats software (programming) commands
- Examples: Arduino Uno, Raspberry Pi, etc.

Acknowledgment: This work is supported by National Science Foundation grant 1749566

1. Meet Arduino ...

- Was developed as a cheap choice for artists, hobbyists, students, and anyone with a gadgetry dream.
- A new challenge: how to teach students to create electronics fast.



http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino

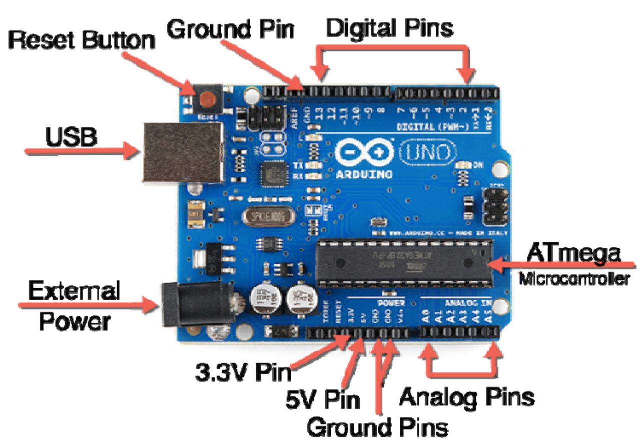
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Our Partner for This Workshop:

https://www.sparkfun.com/sparkfun_inventors_kit
And to access the experiments go to:
https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-experiment-guide---v40

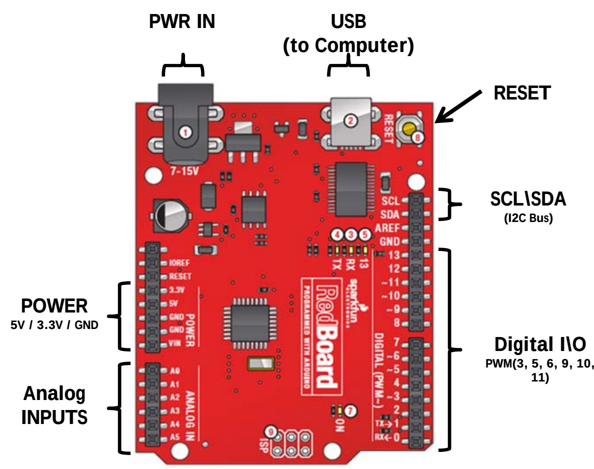
Acknowledgment: This work is supported by National Science Foundation grant 1749566

The Arduino Uno Development Board



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Red Board

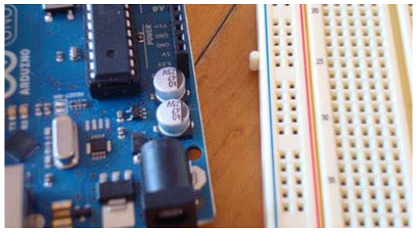


Acknowledgment: This work is supported by National Science Foundation grant 1749566

Arduino & Arduino Compatible Boards

Acknowledgment: This work is supported by National Science Foundation grant 1749566

What is a Development Board?



- A printed circuit board designed to facilitate work with a particular microcontroller.

- Typical components include:
  - Power circuit
  - Programming interface
  - Basic input; usually buttons and LEDs
  - I/O pins

Acknowledgment: This work is supported by National Science Foundation grant 1749566

The Word “Arduino” Can Mean 3 Things

A physical piece of hardware

A programming environment

A community & philosophy

todbot.com/blog/bionicarduino

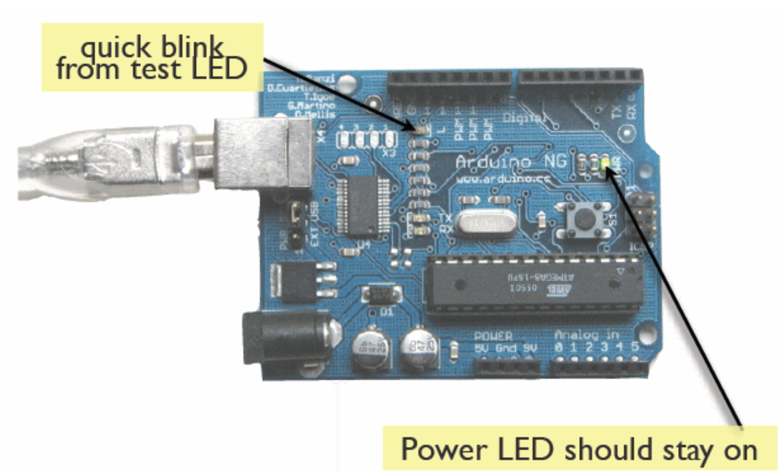
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Getting Started

- Check out: http://arduino.cc/en/Guide/HomePage
- Download & install the Arduino environment (IDE)
- Connect the board to your computer via the UBS cable
- If needed, install the drivers (not needed in lab)
- Launch the Arduino IDE
- Select your board
- Select your serial port
- Open the blink example
- Upload the program

Acknowledgment: This work is supported by National Science Foundation grant 1749566

First Step: Connect the USB Cable



todbot.com/blog/bionicarduino

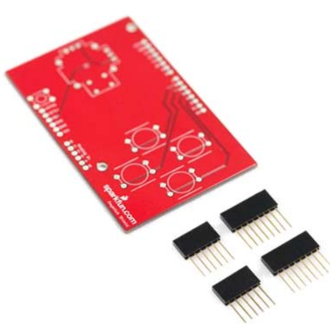
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Go Ahead and Plug Your Board In!

Acknowledgment: This work is supported by National Science Foundation grant 1749566

3. Hardware Components
Arduino Shields:

PCB



Built Shield

Inserted Shield

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Arduino Shields:

Micro SD

MP3 Trigger

LCD

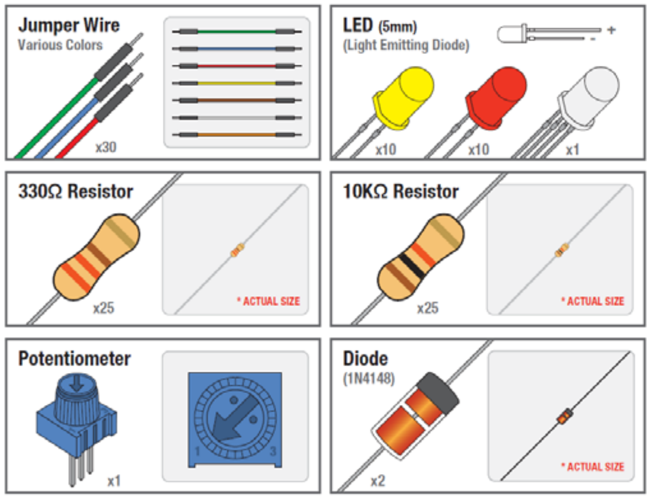
Acknowledgment: This work is supported by National Science Foundation grant 1749566

SIK Components

Push Button	Digital Input	Switch - Closes or opens circuit	Polarized, needs resistor	4 leads
Trim potentiometer	Analog Input	Variable resistor	Also called a Trimpot.	3 leads
Photoresistor	Analog Input	Light Dependent Resistor (LDR)	Resistance varies with light.	3 leads
Relay	Digital Output	Switch driven by a small signal	Used to control larger voltages	3 leads
Temp Sensor	Analog Input	Temp Dependent Resistor		2 leads
Flex Sensor	Analog Input	Variable resistor		3 leads
Soft Trimpot	Analog Input	Variable resistor	Careful of shorts	2 leads
RGB LED	Dig & Analog Output	16,777,216 different colors	Ooh... So pretty.	4 leads

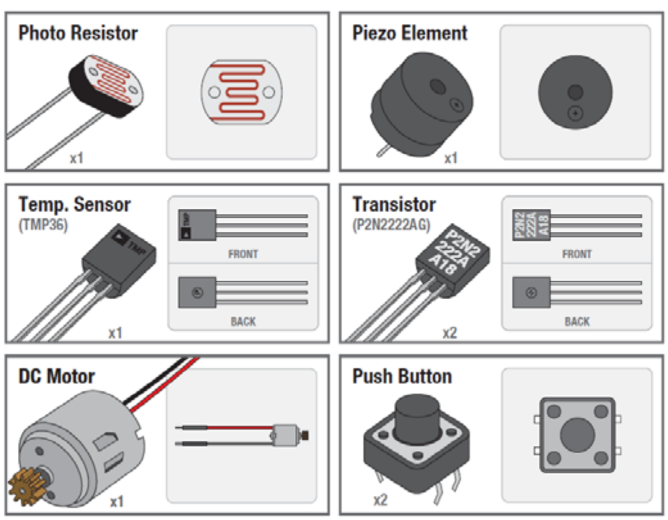
Acknowledgment: This work is supported by National Science Foundation grant 1749566

SIK Components

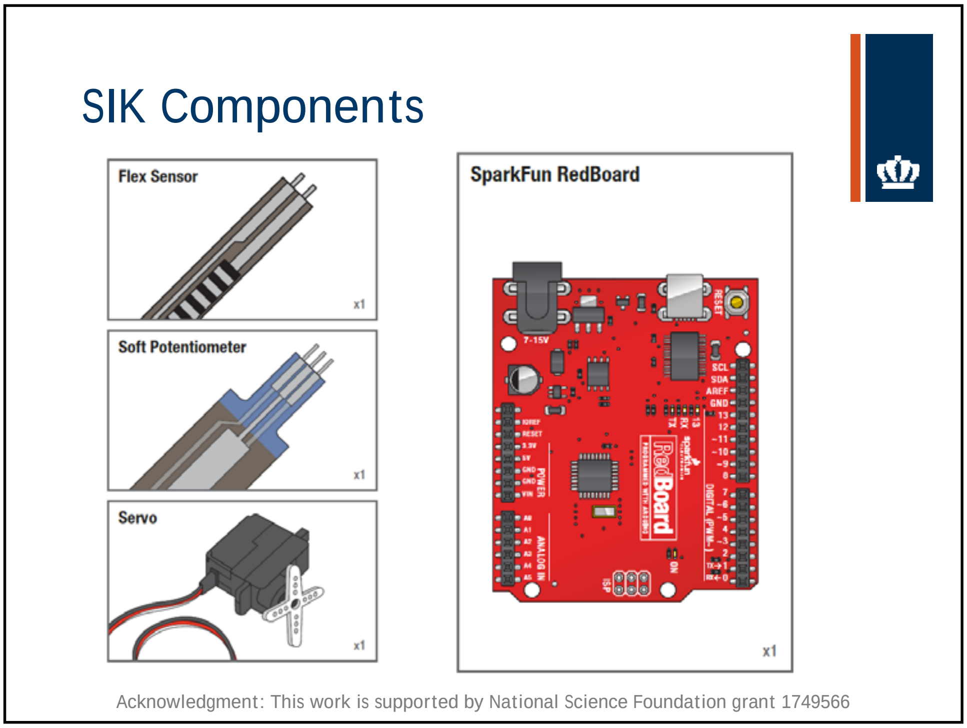


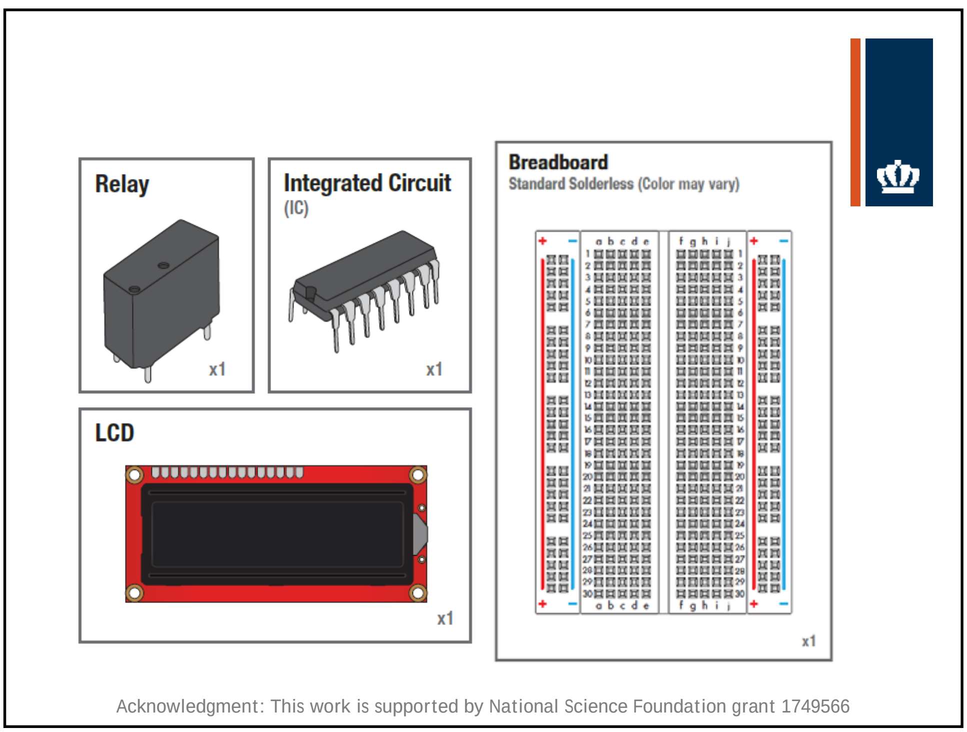
Acknowledgment: This work is supported by National Science Foundation grant 1749566

SIK Components



Acknowledgment: This work is supported by National Science Foundation grant 1749566


SIK Components
Flex Sensor
x1
Soft Potentiometer
x1
Servo
x1
SparkFun RedBoard
x1
Acknowledgment: This work is supported by National Science Foundation grant 1749566


Relay
x1
Integrated Circuit
(IC)
x1
LCD
x1
Breadboard
Standard Solderless (Color may vary)
x1
Acknowledgment: This work is supported by National Science Foundation grant 1749566

3. Basics of Electrical Circuits

- Voltage
- Current
- Resistance
- Ohms Law
- Using a Multi-meter

Some cool presentations about electrical circuits that you can check on your own:

http://bowlesphysics.com/images/05AP_Physics_C_-_Electric_Circuits.ppt

https://www.build-electronic-circuits.com/basic-electronic-components/

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Electrical Properties

Voltage V	Current I	Resistance R
• Defined as the amount of potential energy in a circuit. • Units: Volts (V)	• The rate of charge flow in a circuit. • Units: Amperes (A)	• Opposition to charge flow. • Units: Ohms (Ω)

OHM's LAW: $V = I\ R$

Acknowledgment: This work is supported by National Science Foundation grant 1749566

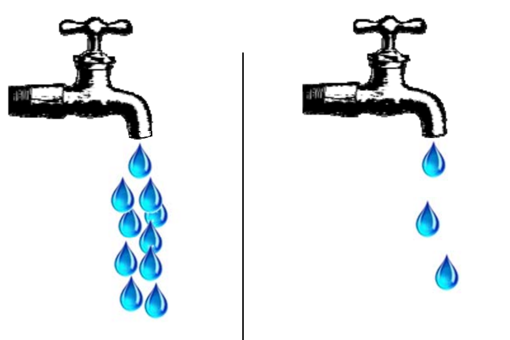
Ohm’s Law

- Ohm’s Law describes the direct relationship between the Voltage (V), Current (I), and Resistance (R) of a circuit.
- The three different forms of Ohm’s Law are as follows

$$V = I \cdot R \quad I = \frac{V}{R} \qquad R = \frac{V}{I}$$

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Current Flow Analogy

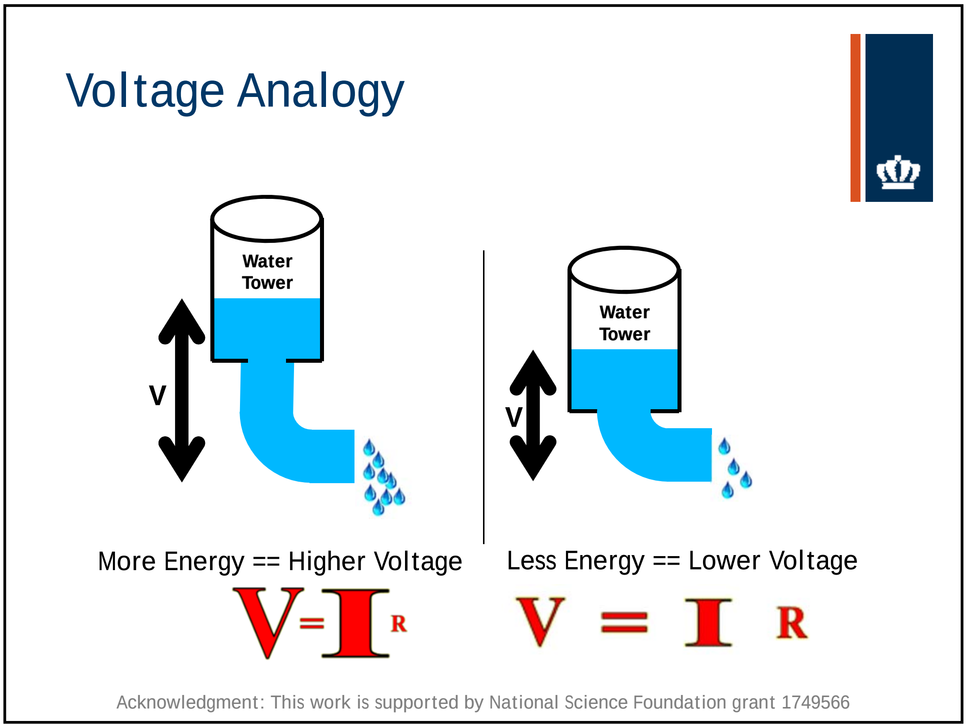


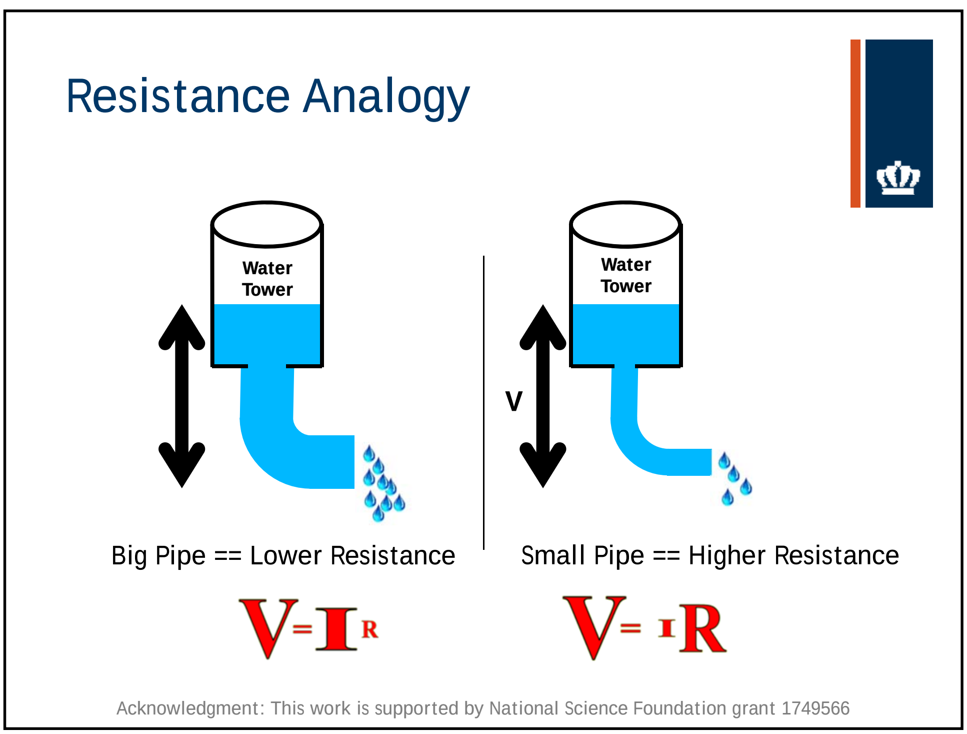
High Current

Low Current

$$V = I\ R$$

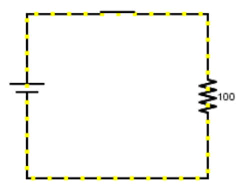
Acknowledgment: This work is supported by National Science Foundation grant 1749566

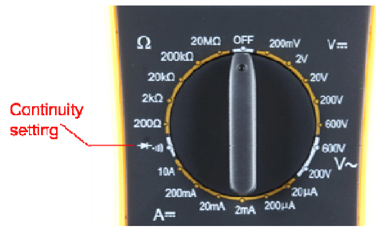

Voltage Analogy
Water Tower
V
Water Tower
V
More Energy == Higher Voltage
Less Energy == Lower Voltage
V=IR
V = I R
Acknowledgment: This work is supported by National Science Foundation grant 1749566


Resistance Analogy
Water Tower
Water Tower
V
Big Pipe == Lower Resistance
Small Pipe == Higher Resistance
V=IR
V= IR
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Continuity – Is it a Circuit?

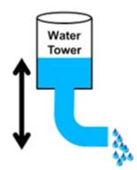
- The word “circuit” is derived from the circle. An Electrical Circuit must have a continuous LOOP from Power (Vcc) to Ground (GND).
- Continuity is important to make portions of circuits are connect. Continuity is the simplest and possibly the most important setting on your multi-meter. Sometimes we call this “ringing out” a circuit.



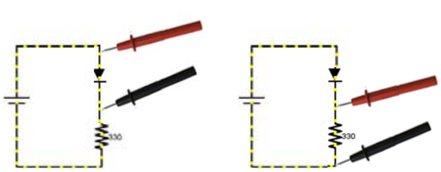


Acknowledgment: This work is supported by National Science Foundation grant 1749566

Measuring Electricity – Voltage

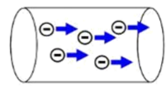


Voltage is a measure of potential electrical energy. A voltage is also called a potential difference – it is measured between two points in a circuit – across a device.



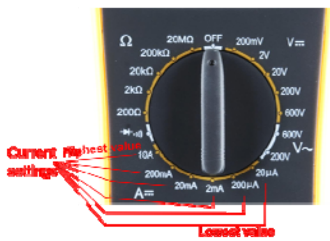
Acknowledgment: This work is supported by National Science Foundation grant 1749566

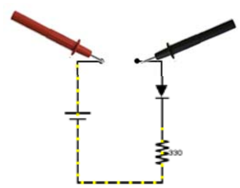
Measuring Electricity -- Current



Current is the measure of the rate of charge flow. For Electrical Engineers – we consider this to be the movement of electrons.

In order to measure this – you must break the circuit or insert the meter <u>in-line</u> (series).

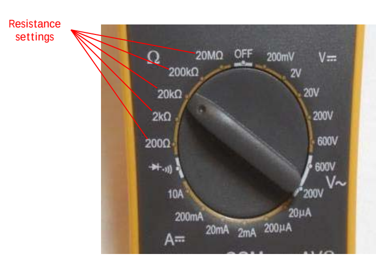




Acknowledgment: This work is supported by National Science Foundation grant 1749566

Measuring Electricity -- Resistance

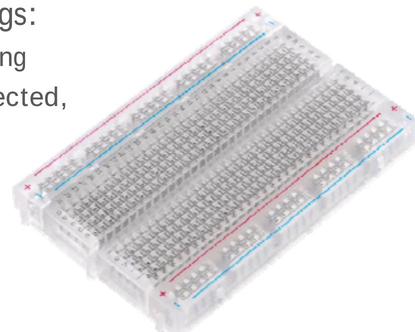
- Resistance is the measure of how much opposition to current flow is in a circuit.
- Components should be removed entirely from the circuit to measure resistance. Note the settings on the multi-meter. Make sure that you are set for the appropriate range.



Acknowledgment: This work is supported by National Science Foundation grant 1749566

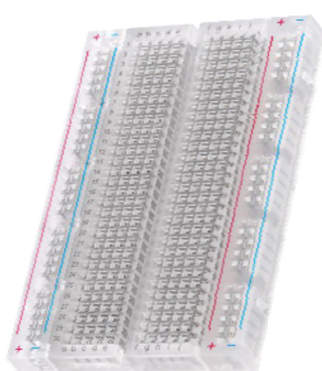
Prototyping Circuits
Solderless Breadboard

- One of the most useful tools in an engineer or Maker’s toolkit.
- The three most important things:
  - A breadboard is easier than soldering
  - A lot of those little holes are connected, which ones?
  - Sometimes breadboards break



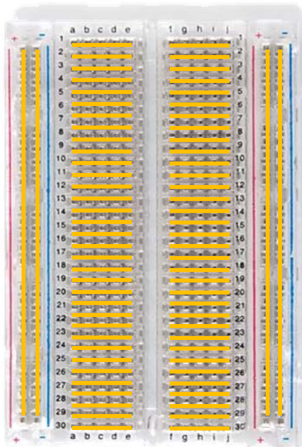
Acknowledgment: This work is supported by National Science Foundation grant 1749566

What is a Breadboard?



Acknowledgment: This work is supported by National Science Foundation grant 1749566

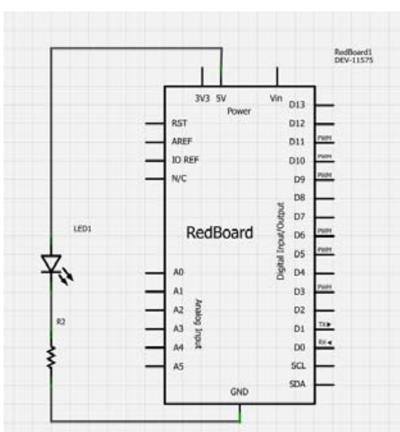
Solderless Breadboard



- Each row (horiz.) of 5 holes are connected.
- Vertical columns – called power bus are connected vertically

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Using the Breadboard to Built a Simple Circuit



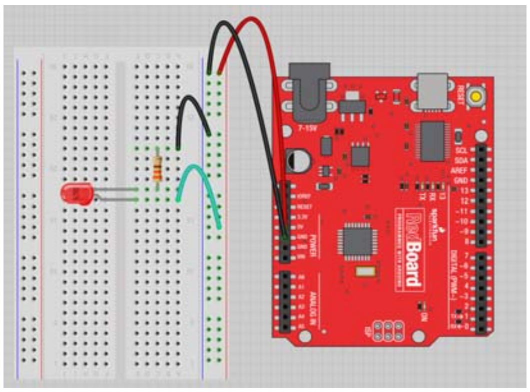
- Use the breadboard to wire up a single LED with a 330 Ohm Resistor (Orange-Orange-Brown).

<u>Note</u>: the longer leg on the LED is the positive leg and the shorter leg is the negative

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Your First Circuit.
Circuit 1A: **BLINK a LED**

- What happens when you break the circuit?
- What if you wanted to add more than one LED?



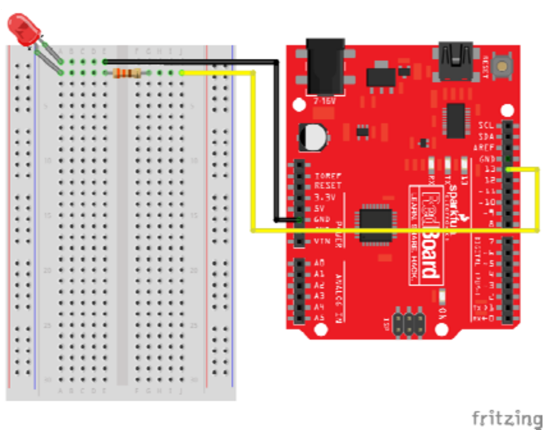
NOTICE:
Follow the connections and notice how the circuit loop closes.

Notice how power rails of the breadboard are used to connect the components to power terminals on the board.

https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-experiment-guide---v40/circuit-1a-blink-an-led

Acknowledgment: This work is supported by National Science Foundation grant 1749566

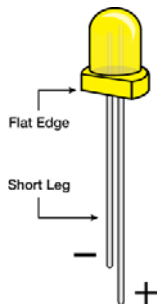
Circuit 1A:
BLINK a LED. Alternative design.



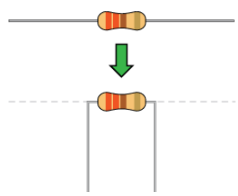
NOTICE:
The power rails of the breadboard are not used here. The circuit closes through PIN 13, used here to power the circuit.

Acknowledgment: This work is supported by National Science Foundation grant 1749566

The components you use in Circuit 1A: BLINK a LED



Light-Emitting Diodes (LEDs) are small lights made from a silicon diode. They come in different colors, brightnesses and sizes. LEDs have a positive (+) leg and a negative (-) leg, and they will only let electricity flow through them in one direction. LEDs can also burn out if too much electricity flows through them, so you should always use a resistor to limit the current when you wire an LED into a circuit.



Resistors

Resistors resist the flow of electricity. You can use them to protect sensitive components like LEDs. The strength of a resistor (measured in ohms) is marked on the body of the resistor using small colored bands.

Acknowledgment: This work is supported by National Science Foundation grant 1749566

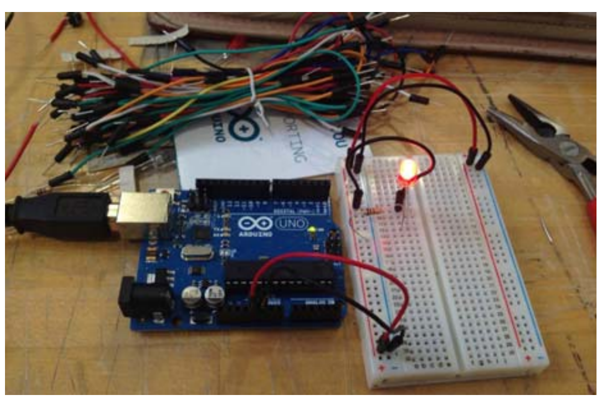
Arduino Module

Intro to Programming

Acknowledgment: This work is supported by National Science Foundation grant 1749566

4. Programming with Arduino

Adding control!



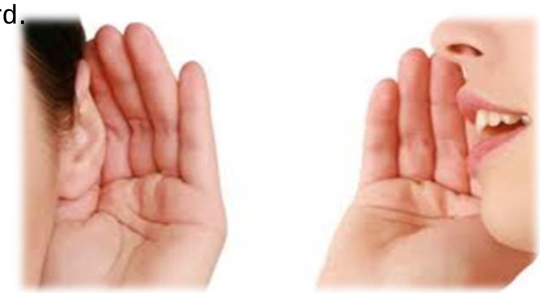
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Concepts: INPUT vs. OUTPUT

Referenced from the perspective of the microcontroller (electrical board).

Inputs is a signal / information going into the board.

Output is any signal exiting the board.



Almost all systems that use physical computing will have some form of output

What are some examples of Outputs?

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Concepts: INPUT vs. OUTPUT

Referenced from the perspective of the microcontroller (electrical board).

Inputs is a signal / information going into the board.

Examples: Buttons Switches, Light Sensors, Flex Sensors, Humidity Sensors, Temperature Sensors…

Output is any signal exiting the board.

Examples: LEDs, DC motor, servo motor, a piezo buzzer, relay, an RGB LED

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Concepts: Analog vs. Digital

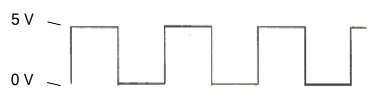
- Microcontrollers are **digital** devices

Signals only take some specific values, sometimes only two values (ON or OFF). Also called – discrete.

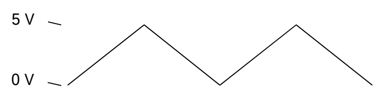
- only values 0 and 5 in the example below

- **Analog** signals can take a full range of values.

- any value between 0 and 5 in the example below



Digital Signal



Analog Signal

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Computers don't really do analog, they **quantize**.

Quantization

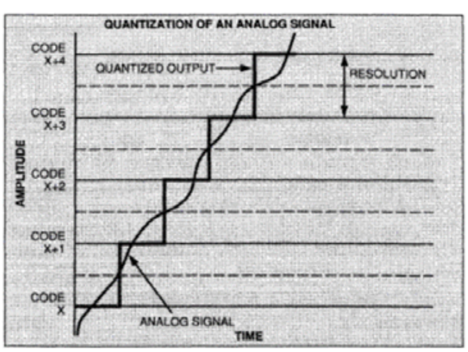
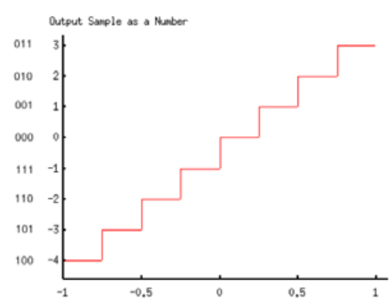
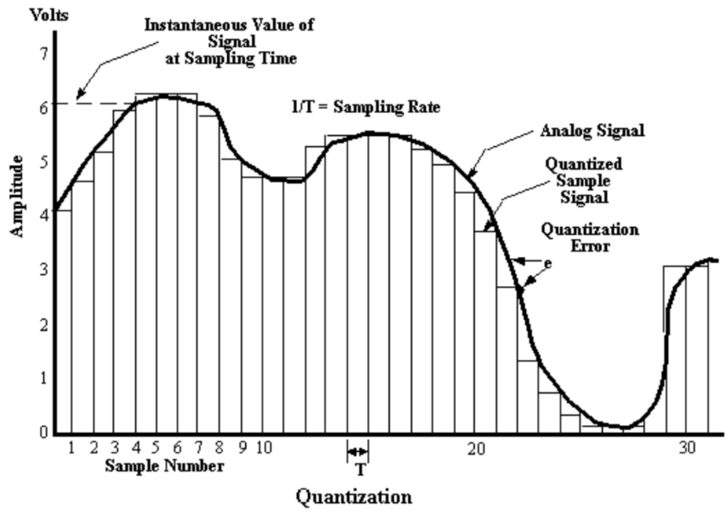


Fig 1—Quantization maps the analog input range into 2^N digital words. This



Acknowledgment: This work is supported by National Science Foundation grant 1749566



Acknowledgment: This work is supported by National Science Foundation grant 1749566

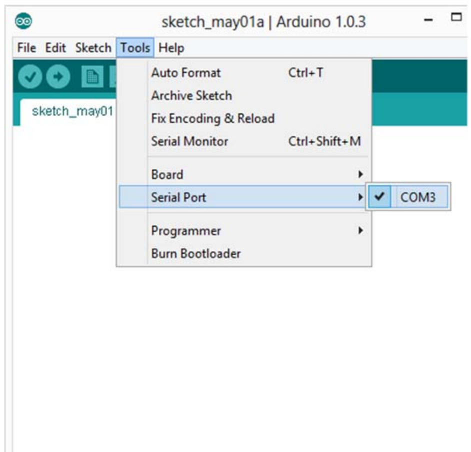
Arduino Integrated Development Environment (IDE)

- Name of current sketch
- Main menus
- Action buttons/icons
  - Verify (AKA compile)
  - Upload (send to Arduino)
  - Start a new sketch
  - Open a sketch (from a file)
  - Save current sketch (to a file)
  - Open Serial Monitor window

Error messages and other feedback show up here.

Acknowledgment: This work is supported by National Science Foundation grant 1749566

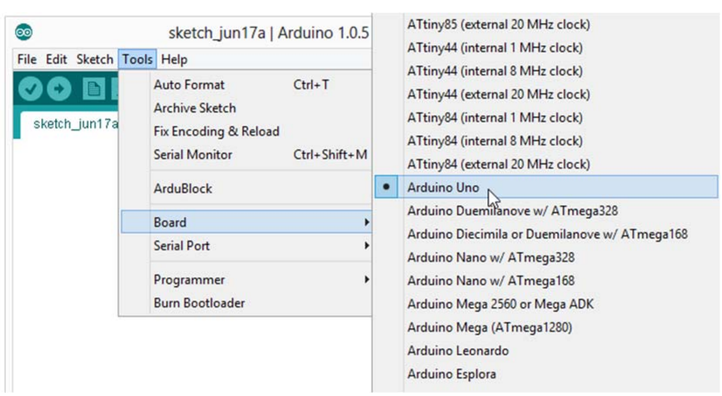
Settings: Tools → Serial Port



- Your computer communicates to the Arduino microcontroller via a serial port → through a USB-Serial adapter.
- Check to make sure that the drivers are properly installed.

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Settings: Tools → Board



- Next, double-check that the proper board is selected under the Tools → Board menu.

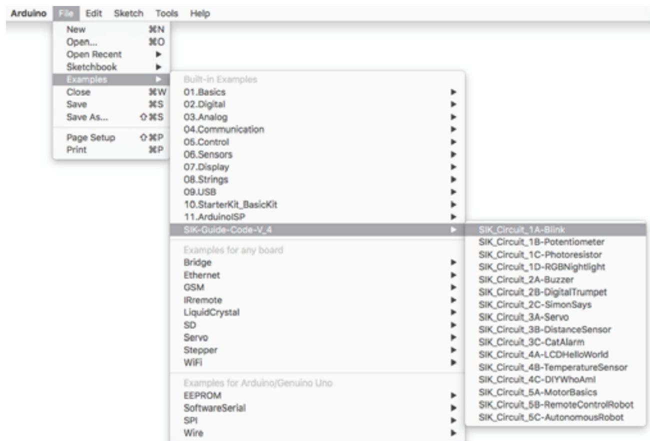
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Open Your First Sketch

Open the Arduino IDE software on your computer. Open the code for Circuit 1A by accessing the SIK Guide Code you downloaded and placed into your examples folder earlier.

To open the code, go to:

File > Examples > SIK_Guide_Code-V_4 > SIK_Circuit_1A-Blink



Acknowledgment: This work is supported by National Science Foundation grant 1749566

A Little Bit About Programming

Two required functions / methods / routines:

```
void setup()
{
        // runs once
}

void loop()
{
        // repeats
}
```

- Code is case sensitive
- Statements are commands and must end with a semi-colon
- Comments follow a // or begin with /* and end with */

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Terminology

"*sketch*" – a program you write to run on an Arduino board

"*pin*" – an input or output connected to something.

e.g. output to an LED, input from a knob.

"*digital*" – value is either HIGH or LOW.

(aka on/off, one/zero) e.g. switch state

"*analog*" – value ranges, usually from 0-255.

e.g. LED brightness, motor speed, etc.

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Commands to Know…

```
pinMode(pin, INPUT/OUTPUT);

    ex: pinMode(13, OUTPUT);//Sets pin to either
                               INPUT or OUTPUT

digitalRead(pin)
  ex: digitalRead(13);// Reads HIGH or LOW from
                         a pin

digitalWrite(pin, HIGH/LOW);

    ex: digitalWrite(13, HIGH);//Writes HIGH
                                 or LOW to a pin

delay(time_ms);

    ex: delay(2500); // delay of 2.5 sec.

// NOTE: -> commands are CASE-sensitive
```

•More commands: arduino.cc/en/Reference/HomePage

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Let’s Get to Coding…

- **Project #1** – Blink
  - “Hello World” of Physical Computing
- *Pseudo-code – how should this work?*

Turn LED ON → Wait → Turn LED OFF → Wait → Rinse & Repeat

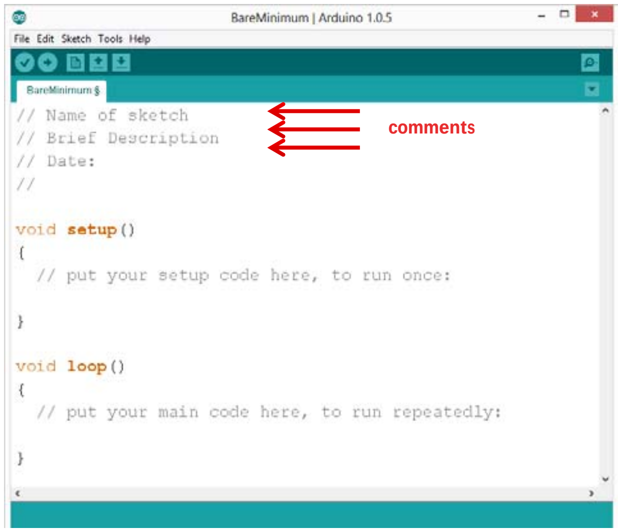
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Comments, Comments, Comments

- Comments are for you – the programmer and your friends…or anyone else human that might read your code.

```
// this is for single line comments
// it’s good to put a description at the top and
before anything ‘tricky’
/* this is for multi-line comments
   Like this…
   And this….
*/
```

Acknowledgment: This work is supported by National Science Foundation grant 1749566



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Arduino Module

Controlling the LED

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Project #1: Wiring Diagram

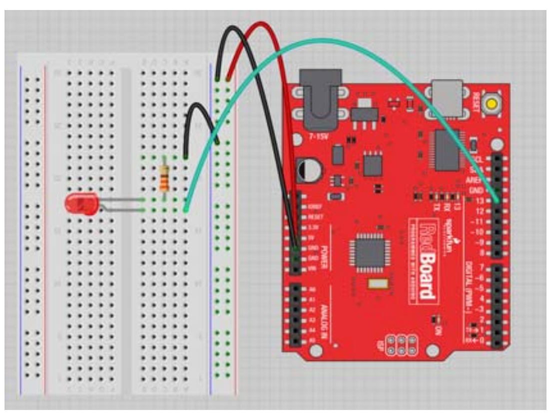


Image created in Fritzing

Move the green wire from the power bus to <u>pin 13</u> (or any other Digital I/O pin on the Arduino board.

Acknowledgment: This work is supported by National Science Foundation grant 1749566

```
/*
SparkFun Inventor's Kit
Circuit 1A-Blink
Turns an LED connected to pin 13 on and off. Repeats forever.
*/

void setup() {

pinMode(13, OUTPUT);      // Set pin 13 to output

}

void loop() {

digitalWrite(13, HIGH);   // Turn on the LED

delay(2000);              // Wait for two seconds

digitalWrite(13, LOW);    // Turn off the LED

delay(2000);              // Wait for two seconds

}
```

The Code of your first circuit.
Circuit 1A:

BLINK a LED

Acknowledgment: This work is supported by National Science Foundation grant 1749566

A few simple challenges
Let's make LED#13 blink!

- **Challenge 1a** – blink with a 200 ms second interval.
- **Challenge 1b** – blink to mimic a heartbeat
- **Challenge 1c** – find the fastest blink that the human eye can still detect…

1 ms delay? 2 ms delay? 3 ms delay???

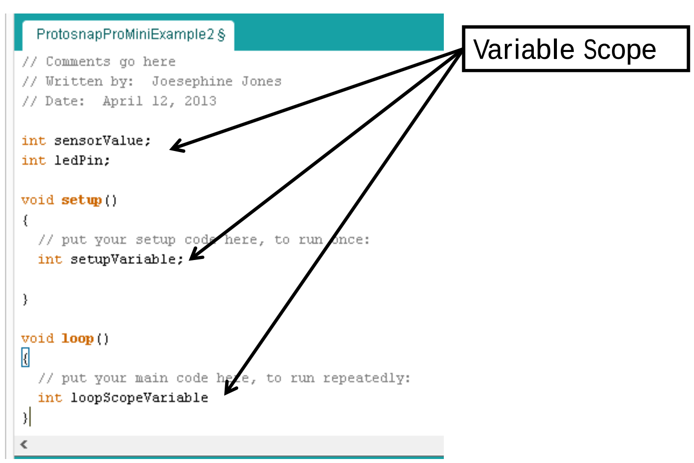
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Try adding other LEDs

- Can you blink two, three, or four LEDs?
  (Hint: Each LED will need it's own 330Ω resistor.)
- Generate your own morse code flashing
- How about → Knight Rider? Disco? Police Light?

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Programming Concepts: Variables



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Programming Concepts: Variable Types

- **Variable Types:**

8 bits	16 bits	32 bits
byte char	int unsigned int	long unsigned long float

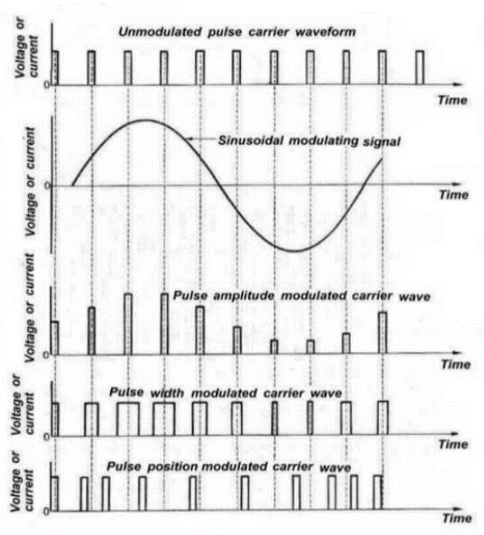
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Fading in and Fading Out (Analog or Digital?)

- A few pins on the Arduino allow for us to modify the output to mimic an analog signal.
- This is done by a technique called:

  Pulse Width Modulation (PWM)

Acknowledgment: This work is supported by National Science Foundation grant 1749566



PULSE MODULATION

PAM

PWM

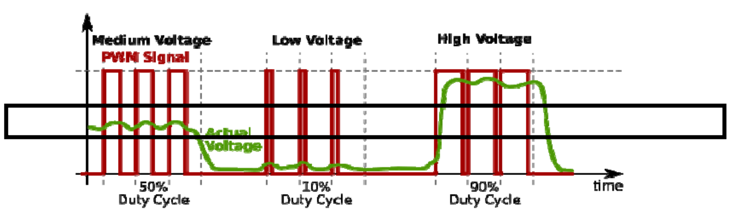
PPM

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Concepts: Analog vs. Digital

■To create an analog signal, the microcontroller uses a technique called PWM. By varying the duty cycle, we can mimic an "average" analog voltage.

Pulse Width Modulation (PWM)



Acknowledgment: This work is supported by National Science Foundation grant 1749566

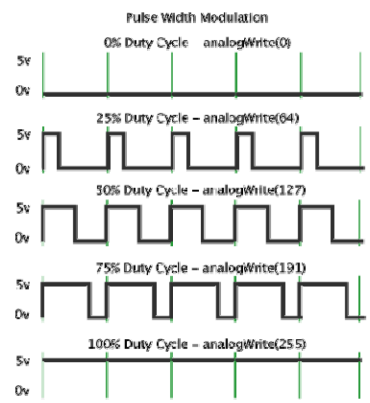
Project #2 – Fading
Introducing a new command…

```
analogWrite(pin, val);
```

pin – refers to the OUTPUT pin (limited to pins 3, 5, 6, 9, 10, 11.) – denoted by a ~ symbol

val – 8 bit value (0 – 255).

0 => 0V | 255 => 5V

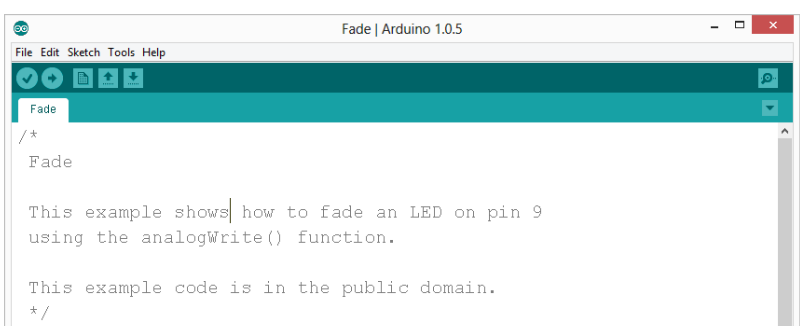


Acknowledgment: This work is supported by National Science Foundation grant 1749566

Move one of your LED pins over to Pin 9

- In Arduino, open up:

File → Examples → 01.Basics → Fade



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Fade - Code Review

Fade | Arduino 1.0.5

File Edit Sketch Tools Help

Fade

```
/*
 Fade

 This example shows how to fade an LED on pin 9
 using the analogWrite() function.

 This example code is in the public domain.
 */

int led = 9;           // the pin that the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by
```

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Fade - Code Review

```
void setup()  {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop()  {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

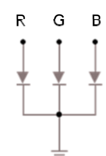
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Project# 2 -- Fading

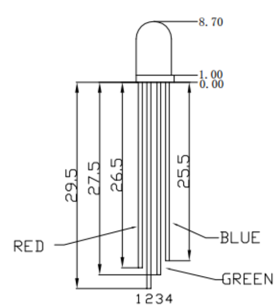
- **Challenge 2a** – Change the rate of the fading in and out. There are at least two different ways to do this – can you figure them out?
- **Challenge 2b** – Use 2 (or more) LEDs – so that one fades in as the other one fades out.

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Color Mixing
Tri-color LED

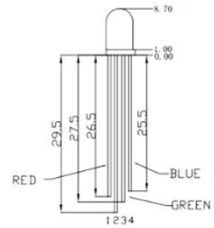


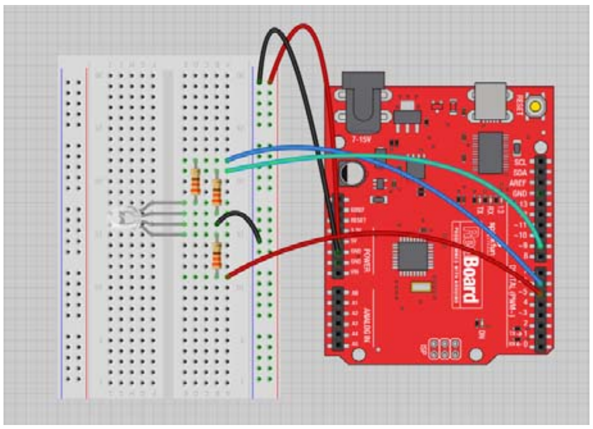
- In the SIK, this is a standard – Common Cathode LED
- This means the negative side of the LED is all tied to Ground.



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Project 3 – RGB LED





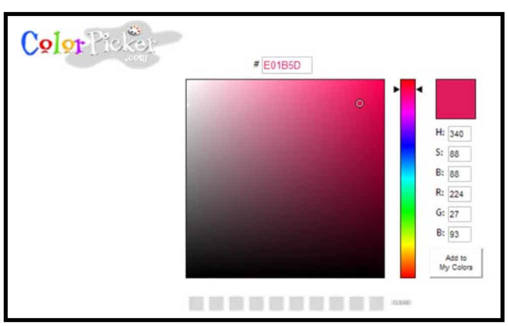
Note: The longest leg of the RGB LED is the Common Cathode. This goes to GND.

Use pins 5, 6, & 9

Acknowledgment: This work is supported by National Science Foundation grant 1749566

How Many Unique Colors Can You Create?

$$\# \textit{ of unique colors} = 256 \cdot 256 \cdot 256$$
$$= 16{,}777{,}216 \textit{ colors!}$$



Use Colorpicker.com or experiment on your own.

Pick out a few colors that you want to try re-creating for a lamp or lighting display...

Play around with this with the `analogWrite()` command.

Acknowledgment: This work is supported by National Science Foundation grant 1749566

RGB LED Color Mixing

```
int redPin = 5;
int greenPin = 6;
int bluePin = 9;

void setup()
{
        pinMode(redPin, OUTPUT);
        pinMode(greenPin, OUTPUT);
        pinMode(bluePin, OUTPUT);
}
```

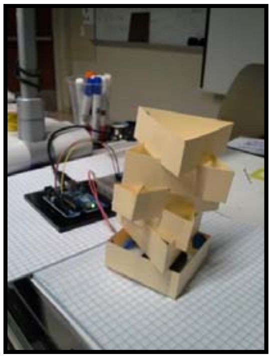
Acknowledgment: This work is supported by National Science Foundation grant 1749566

RGB LED Color Mixing

```
void loop()
{
        analogWrite(redPin, 255);
        analogWrite (greenPin, 255);
        analogWrite (bluePin, 255);
}
```

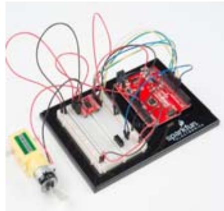
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Project: Mood Lamp / Light Sculpture



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Arduino Module



Controlling Motors

Acknowledgment: This work is supported by National Science Foundation grant 1749566

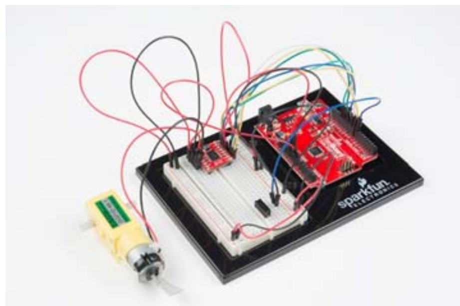
About Motors

- There are several different types
- Standard DC motor - input current for full continuous rotation. No special pins or wiring.
- Standard servomotor (AKA servos) - Motor capable of limited rotation (generally 180°) in precise degree increments. Uses Servo library in Arduino. Have 3+ pins. Controlled by pulse-width modulation ("~" pins)
- Continuous rotation servo – can go all the way around continuously. Interprets PWM value as speed & dir.
- Stepper Motors - Servo capable of full rotation in small steps. Uses Stepper library in Arduino. Have 3+ pins
- We are using a standard servo in this lesson.

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Circuit 5A: Motor Basics

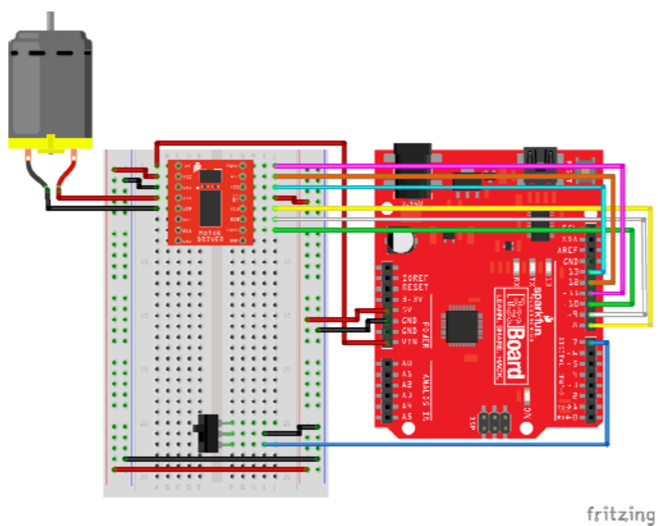
- In this circuit you will learn the basic concepts behind motor control. Motors require a lot of current, so you can't drive them directly from a digital pin on the RedBoard.
- Instead, you'll use what is known as a motor controller or motor driver board to power and spin the motor accordingly.



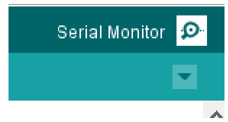
https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-experiment-guide---v40/circuit-5a-motor-basics

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Circuit Diagram:



Acknowledgment: This work is supported by National Science Foundation grant 1749566




```
/*
SparkFun Inventor's Kit
Circuit 5A - Motor Basics

Learn how to control one motor with the motor driver.
*/

//PIN VARIABLES
//the motor will be controlled by the motor A pins on the motor driver
const int AIN1 = 13;          //control pin 1 on the motor driver for the right
motor
const int AIN2 = 12;          //control pin 2 on the motor driver for the right
motor
const int PWMA = 11;          //speed control pin on the motor driver for
the right motor

int switchPin = 7;            //switch to turn the robot on and off

//VARIABLES
int motorSpeed = 0;       //starting speed for the motor
```

Acknowledgment: This work is supported by National Science Foundation grant 1749566

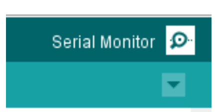
Open Serial Monitor

```
void setup() {
  pinMode(switchPin, INPUT_PULLUP);   //set this as a pullup to sense
whether the switch is flipped

  //set the motor contro pins as outputs
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);
  pinMode(PWMA, OUTPUT);

  Serial.begin(9600);       //begin serial communication with the computer

  Serial.println("Enter motor speed (0-255)... ");
  //Prompt to get input in the serial monitor.
}
```



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Enter Motor Speed 0 - 255

```
void loop() {
  if (Serial.available() > 0){          //if the user has entered something in the serial monitor
    motorSpeed = Serial.parseInt();     //set the motor speed equal to the number in the serial message

    Serial.print("Motor Speed: ");      //print the speed that the motor is set to run at
    Serial.println(motorSpeed);
  }

  if(digitalRead(7) == LOW){            //if the switch is on...
      spinMotor(motorSpeed);
  } else{                               //if the switch is off...
      spinMotor(0);                     //turn the motor off
  }

}
```

Acknowledgment: This work is supported by National Science Foundation grant 1749566

```
/************************************************/
void spinMotor(int motorSpeed)                    //function for driving the right motor
{
  if (motorSpeed > 0)                             //if the motor should drive forward (positive speed)
  {
    digitalWrite(AIN1, HIGH);                     //set pin 1 to high
    digitalWrite(AIN2, LOW);                      //set pin 2 to low
  }
  else if (motorSpeed < 0)                        //if the motor should drive backwar (negative speed)
  {
    digitalWrite(AIN1, LOW);                      //set pin 1 to low
    digitalWrite(AIN2, HIGH);                     //set pin 2 to high
  }
  else                                            //if the motor should stop
  {
    digitalWrite(AIN1, LOW);                      //set pin 1 to low
    digitalWrite(AIN2, LOW);                      //set pin 2 to low
  }
  analogWrite(PWMA, abs(motorSpeed));             //now that the motor direction is set, drive it at the entered speed
}
```

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Programming: Conditional Statements if()

If this is TRUE…

```
if (analogValue > threshold) {
  digitalWrite(ledPin, HIGH);
}
else {
  digitalWrite(ledPin, LOW);
}
```

Do this.

Otherwise, do this.

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Servo Motors

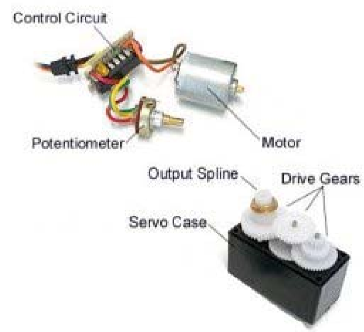
- This little motor is high in efficiency and power
- Servo motors are small in size but are powerful enough and are very energy-efficient.
- These features allow them to be used to operate remote-controlled or radio-controlled toy cars, robots and airplanes.
- Servo motors are also used in industrial applications, robotics, in-line manufacturing, pharmaceutics and food services.

https://www.jameco.com/jameco/workshop/howitworks/how-servo-motors-work.html

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Servo Motors - How does the it work?

- The servo circuitry is built right inside the motor unit and has a positionable shaft, which usually is fitted with a gear.
- The motor is controlled with an electric signal which determines the amount of movement of the shaft.

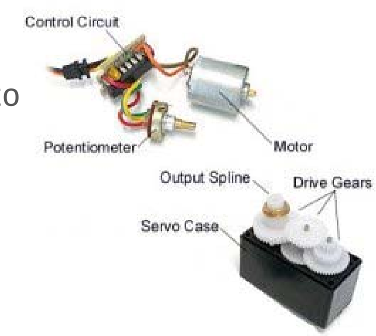


https://www.jameco.com/jameco/workshop/howitworks/how-servo-motors-work.html

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Servo Motors - What is inside?

- A small DC motor, a potentiometer, and a control circuit.
- The motor is attached by gears to the control wheel. As the motor rotates, the potentiometer's resistance changes, so the control circuit can precisely regulate how much movement there is and in which direction.

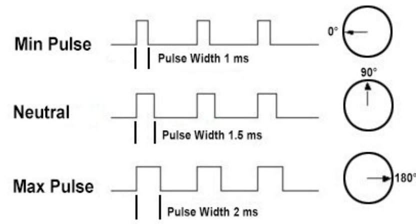


https://www.jameco.com/jameco/workshop/howitworks/how-servo-motors-work.html

Acknowledgment: This work is supported by National Science Foundation grant 1749566

How is a Servo Motor Controlled?

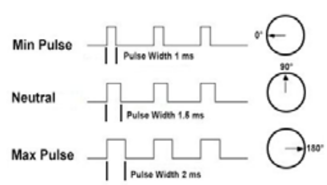
- When the shaft of the motor is at the desired position, power supplied to the motor is stopped.
- If not, the motor is turned in the appropriate direction.
- The desired position is sent via electrical pulses through the signal wire.
- The motor's speed is proportional to the difference between its actual position and desired position.
- So if the motor is near the desired position, it will turn slowly, otherwise it will turn fast. This is called proportional control.



Acknowledgment: This work is supported by National Science Foundation grant 1749566

How Is Servo Controlled?

- Servos are controlled by sending an electrical pulse of variable width, or pulse width modulation (PWM), through the control wire.
- There is a minimum pulse, a maximum pulse, and a repetition rate.
- The PWM sent to the motor determines position of the shaft, and based on the duration of the pulse sent via the control wire; the rotor will turn to the desired position.
- The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns.



Acknowledgment: This work is supported by National Science Foundation grant 1749566

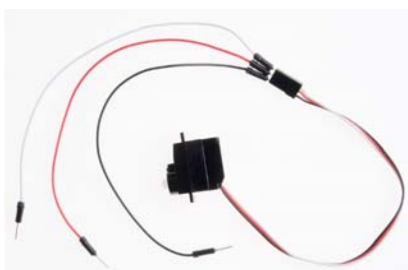
Circuit 3A: Servo Motors

- Regular DC motors have two wires. When you hook the wires up to power, the motor spins around and around.
- **Servo Motors**, on the other hand, have three wires: one for power, one for ground and one for signal. When you send the right signal through the signal wire, the servo will move to a specific angle and stay there. Common servos rotate over a range of 0° to 180°. The signal that is sent is a PWM signal, the same used to control the RGB LED project.

https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-experiment-guide---v40/circuit-3a-servo-motors

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Connecting the Servo Motor:

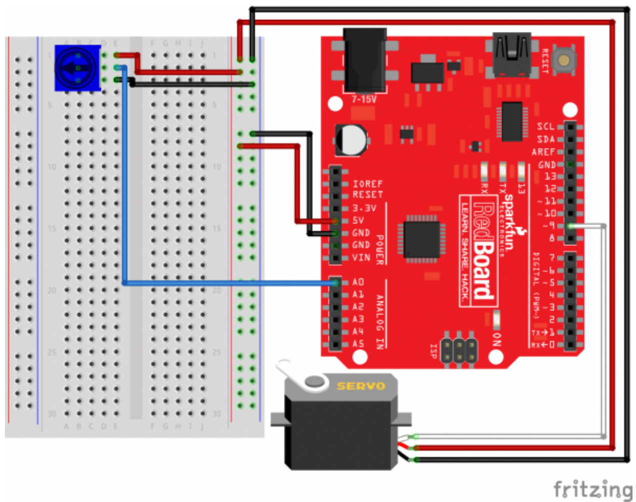


The servo wires are color coded to make hookup simple.
The pin-out is as follows:

Pin	Description
White	Signal - PWM In
Red	Power (5V)
Black	Ground (GND)

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Circuit Diagram:



Acknowledgment: This work is supported by National Science Foundation grant 1749566

```
/*SparkFun Inventor's Kit
Circuit 3A-Servo
Move a servo attached to pin 9 so that it's
angle matches a potentitometer attached
to A0.*/

#include <Servo.h>          //include the
servo library

int potPosition;          //this variable will
store the           position of the
potentiometer
int servoPosition;          //the servo will
move to this          position

Servo myservo;          //create a servo
object

void setup() {

  myservo.attach(9);          //tell the servo
object that its          servo is
plugged into pin 9

}

void loop() {

  potPosition = analogRead(A0);
//use analog read      to measure the
position of the potentiometer (0-1023)

  servoPosition = map(potPosition,
0,1023,20,160);  //convert the
      potentiometer number to a servo
position from 20-160

  myservo.write(servoPosition);
//move the servo to   the 10 degree
position
}
```

Acknowledgment: This work is supported by National Science Foundation grant 1749566

analogRead()

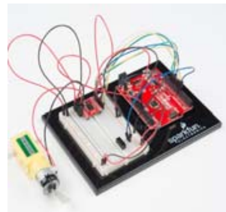
- Arduino uses a 10-bit A/D Converter:

- This means that you get input values from 0 to 1023
  - 0 V → 0
  - 5 V → 1023

For example:

int sensorValue = analogRead(A0)

Acknowledgment: This work is supported by National Science Foundation grant 1749566




Arduino Module

Controlling Multiple Motors

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Using Servo Motors to Control the Movement of Bio-Inspired Robots

- Use servo Motors for positioning

```
#include <Servo.h>
Servo myservo;
void setup()
{
      myservo.attach(9);
      myservo.write(90); // set servo to
mid-point . Range: 0 – 180
 }
void loop()
{
     int theta = 30;
}
```

Pulse Train for
Theta = 30 degrees

Acknowledgment: This work is supported by National Science Foundation grant 1749566

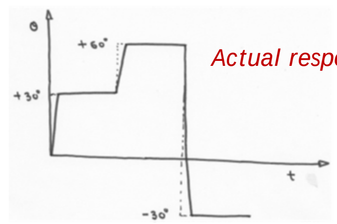
Multi-Step Gait:

```
#include <Servo.h>
Servo myservo;
void setup()
{
myservo.attach(9);
myservo.write(90); // set servo to mid-point .
}
void loop()
{
int theta = 30;
delay(10000);

theta = 60;
delay(10000);

theta = -30;
delay(10000);
}
```

Desired response



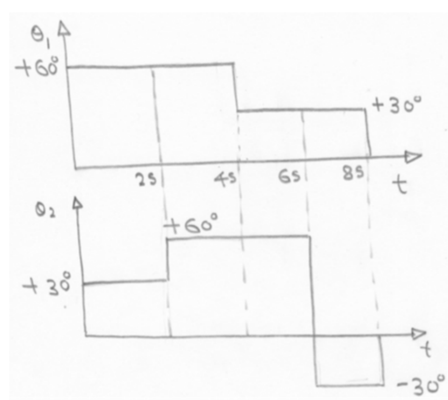
Actual response

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Multi-Motor Gait:

```
#include <Servo.h>
Servo myservo1, myservo2;
void setup()
{
      myservo1.attach(9);
      myservo2.attach(10);
      myservo1.write(90); // set servo to
mid-point.      myservo2.write(90); // set
servo to mid-point.
}
void loop()
{
      int theta1 = 60;
      int theta2 = 30;
      delay(2000);

      theta2 = 60
      delay(2000);
}
```



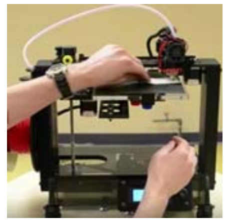
Acknowledgment: This work is supported by National Science Foundation grant 1749566

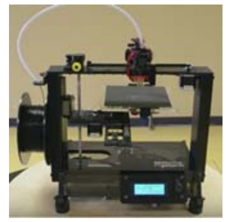
References:

- Sparkfun Electronics – Intro to Arduino
- https://create.coloradovirtuallibrary.org/wp-content/.../SparkFun/.../IntrotoArduino.ppt
- www.patarnott.com/atms360/Arduino/presentations/ArduinoPart1.ppt

Acknowledgment: This work is supported by National Science Foundation grant 1749566

3D Printing





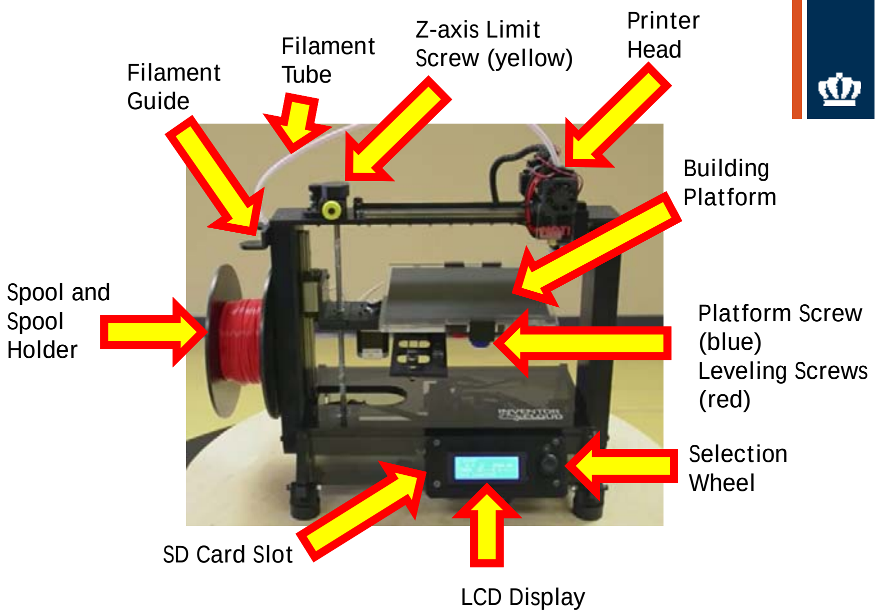
Module Content:

- 3D Printing Intro
- 3D Printing Technologies: FDM and PJ

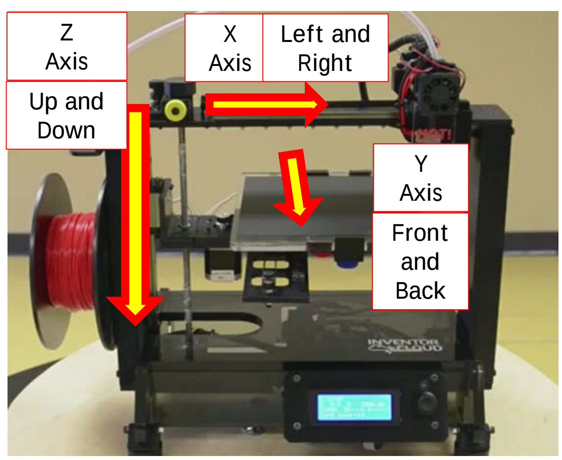
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Invent 3D Printer

Invent 3D Printer



Invent 3D Printer



WARNING Z Axis is locket by default, unlock the Yellow Z axis screw

WARNING Change default step to 1 mm, default is 100 mm > it moves very fast!

[Unclassified]

Invent 3D Printer – Material

PLA – poly lactic acid

- Thermoplastic polymer - softens when heated and hardens when cooled
- Made from renewable resources - corn starch and sugarcane
- Biodegradable - turned into fertilizer by microbes under the right conditions, and must reach 140 degrees for 10 days, in order to be composted.

Invent 3D Printer – Material

PLA – poly lactic acid

- Extruded at 225ºC
- Characteristics similar to:
  - polypropylene (PP) – living hinges (cap in tic tacs)
  - polyethylene (PE) – most common plastic (bottles and plastic bags)
- Used in the Invent 3D Printer as both main material and support material (break away)

PLA – Properties

Melt Temperature	157 - 170 °C (315 - 338 °F)
Typical Injection Molding Temperature	178 - 240 °C (353 - 464 °F)
Heat Deflection Temperature (HDT)	49 - 52 °C (121 - 126 °F) at 0.46 MPa (66 PSI)
Tensile Strength	61 - 66 MPa (8840 - 9500 PSI)
Flexural Strength	48 - 110 MPa (6,950 - 16,000 PSI)
Specific Gravity	1.24
Shrink Rate	0.37 - 0.41% (0.0037 - 0.0041 in/in)

Let’s Send a Part to be 3D Printed

3D Printing

Terminology

Rapid Prototyping (RP)

Solid Freeform Fabrication (SFF)

Additive Manufacturing (AM)

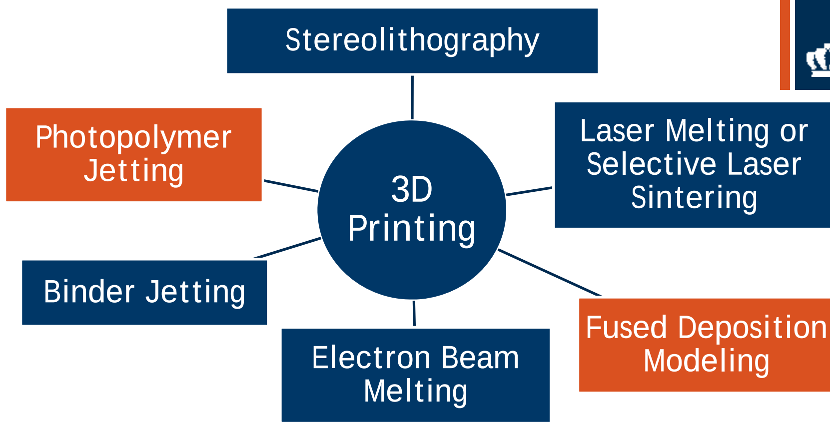
Direct Digital Manufacturing (DDM)

3D Printing

- Layered Fabrication
- Bottom-up fabrication approach
- Distinct from subtractive manufacturing

Timeline

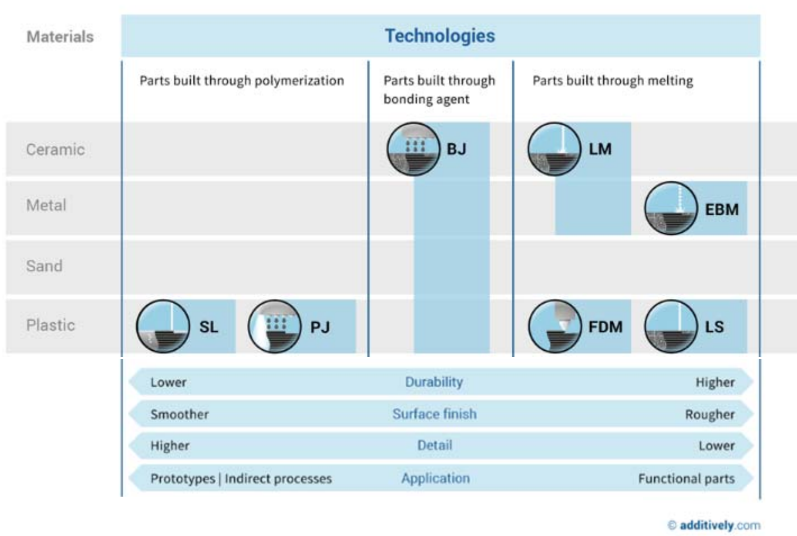
3D Printing Technologies Overview

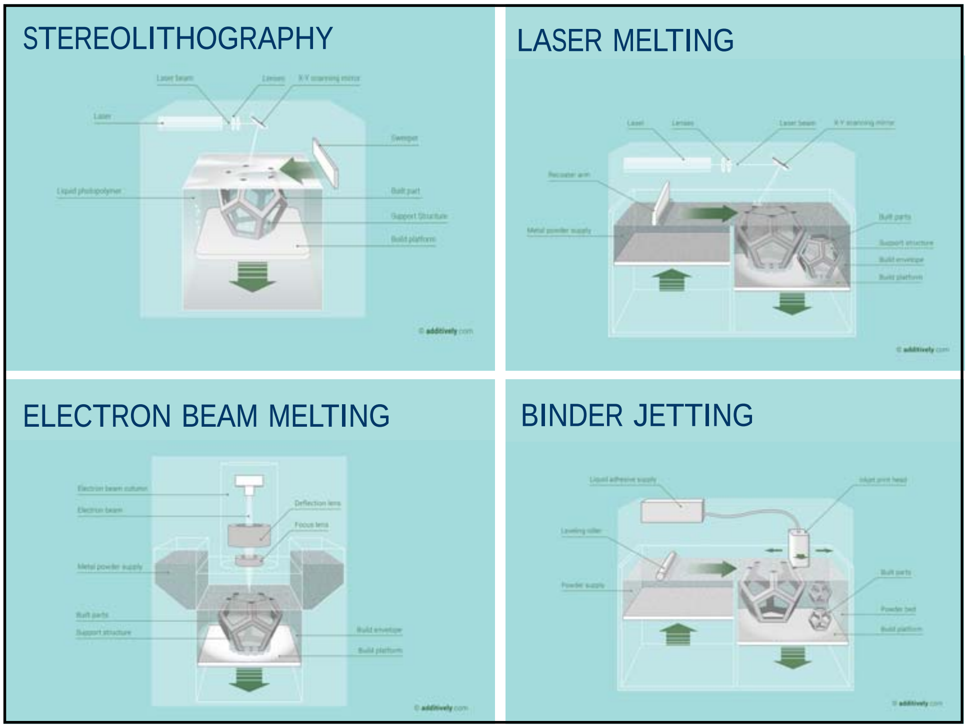


Commonalities:
- Building Envelop (building platform)
- Layered Fabrication
- Bottom-up fabrication approach

Many others

Overview of AM Technologies





STEREOLITHOGRAPHY

- Laser in the UV range, with power of 5 – 30 W
- Materials: Photoreactive Polymers (thermosets)

LASER MELTING

- Laser with 50 – 1000 W of power
- Materials in powder form:
  - Metal alloys
  - Ceramics
  - Polymers

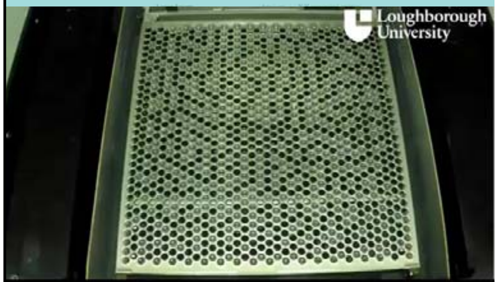
ELECTRON BEAM MELTING

- Electron Beam with 30 – 45 kW of power
- Materials in powder form:
  - Metal alloys
  - Metals

BINDER JETTING

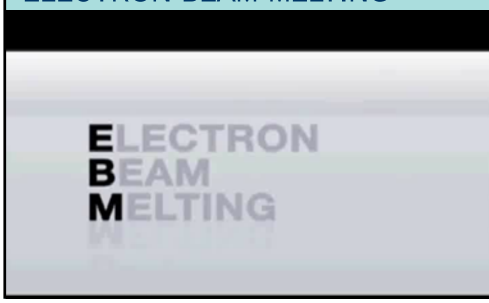
- Adhesive
- Any materials in powder form (metals, ceramics, plastic, sand, gypsum)

STEREOLITHOGRAPHY



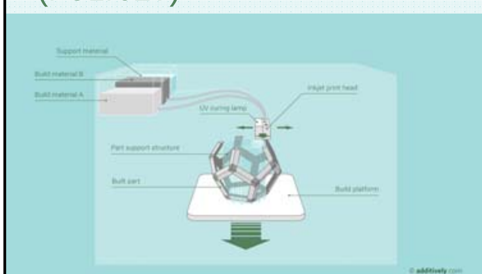
LASER MELTING

ELECTRON BEAM MELTING



BINDER JETTING

PHOTOPOLYMER JETTING (POLYJET)



ADVANTAGES

- Multiple materials can be jetted together allowing multi-material and multi-color parts
- Functionally graded materials are possible
- Can achieve good accuracy and surface finishes

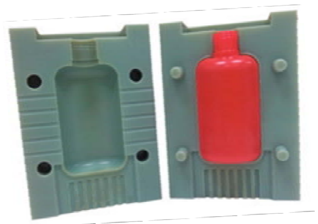
DISADVANTAGES

- Does not work with standard materials but with UV-active photopolymers which are not durable over time (thermoset)
- Parts have limited mechanical properties

PHOTOPOLYMER JETTING (POLYJET)

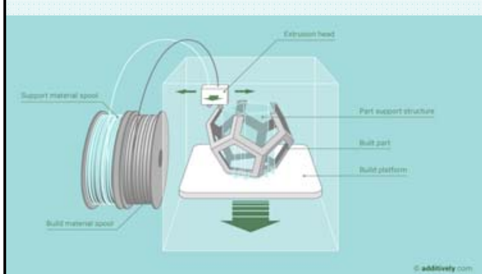
APPLICATIONS

- **Prototypes**
- **Casting patterns**
- **Tools for injection molding**



https://youtu.be/Som3CddHfZE

FUSED-DEPOSITION-MODELING (FDM)

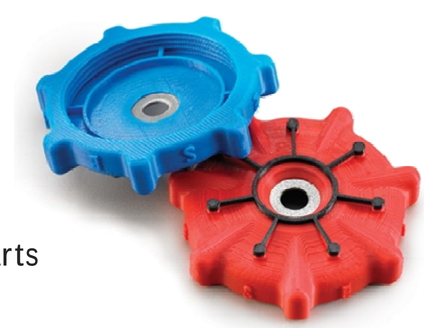


ADVANTAGES

- Parts have good mechanical properties and are durable over time.
- Can build fully functional parts in standard plastics.
- Parts can be post-processed

DISADVANTAGES

- Step structure on surfaces
- Fine details cannot be realized



FUSED DEPOSITION MODELING (FDM)

APPLICATIONS

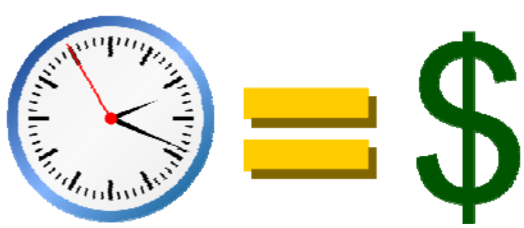
- **Prototypes**
- **Spare parts**
- **Small series parts**




https://youtu.be/WHO6G67GJbM

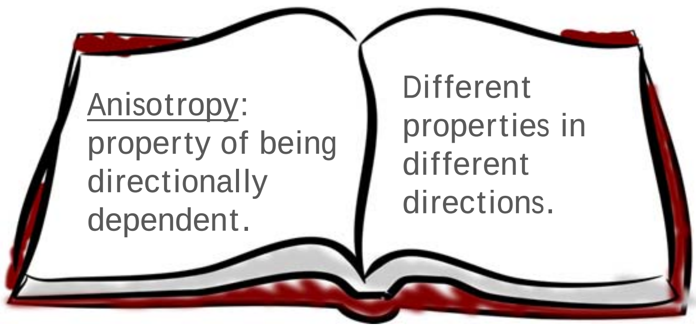
Main Advantage of AM

- Reducing time for development of new products
- Reducing manufacturing time



Main Disadvantage of AM

- Anisotropy in the z-direction (vertical direction)



Example: **wood**, its properties vary when gauged with or against the grain.

AM in Construction

March, 2017

March, 2018

AM in the Olympics

www.3ders.org
3D printer and 3D printing news

Home | Price Compare | Videos | Stats | 3D Printing Basics | Forums

Winter Olympics: 3D printing helps France's greatest Olympian scoop biathlon gold

Feb 22, 2018 | By Benedict

Frenchman Martin Fourcade, winner of five biathlon gold medals at the Winter Olympics, has been shooting his way to victory using a rifle made with 3D printing technology. The rifle was developed by Athletics 3D – Olympics & Fitting and prototyped on a Zortrax 3D printer.

Martin Fourcade shooting his way to victory and Olympic history

(Image: Christian Bier)

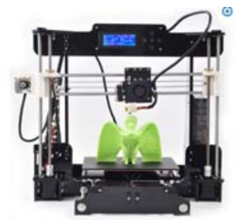
Current State of AM

- AM for production quantities
  - Aerospace
  - Medical
  - Dentistry

Great fit – **production quantities** are relatively **low** but **complexity and value** are usually **high**

Current State of AM

- Proliferation of desktop printers
  - Under $5K with average selling price $1,055 – many educational institutions using them for hands-on learning



Current State of AM

- Large organizations are using AM for production volumes:

AIRBUS

JABIL
(30% of parts could be 3D printed in the future)

UNDER ARMOUR.

ASTON MARTIN

LOCKHEED MARTIN

stryker®

HONDA
The Power of Dreams

Reebok

Current State of AM

- Challenges:
  - Cost of industrial machines and materials for production applications
  - Understanding design for additive manufacturing (DfAM) in the current workforce – less than 1% of practicing engineers and designers have the appropriate knowledge and skills
  - Postprocessing methods and automation know-how

Current State of AM

- Challenges:
  - Future workforce development:
    - Experience in 3D content creation, design optimization, design and process analysis simulation
    - Serve in applications development, R&D, materials science, process controls, part inspection, IT software development, cloud services, materials handling, and automation
  - Create quality education and training programs in:
    - Part-building
    - AM machine operation and maintenance

Want to know more?

Check the movie
"Print the Legend"

References

- Additively.com
- Stratasys Education Open Curriculum Program Available at: http://www.stratasys.com/industries/education/educators/curriculum/introduction-to-3d-printing
- Wohlers Report 2017
- sme.org/smartmfgseries 2016

References to Videos

FDM and Polyjet https://www.cnbc.com/video/2018/04/04/how-3-d-printing-is-helping-to-transform-formula-one-racing.html

Stereolithography https://youtu.be/oNpAnBhgIIs

Laser Melting https://youtu.be/9E5MfBAV_tA?t=15s

Binder Jetting https://youtu.be/RNNxEoXuvuw

Photopolymer Jetting https://youtu.be/Som3CddHfZE

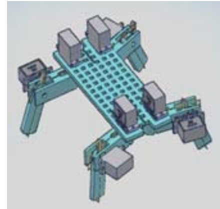
Fused Deposition Modeling https://youtu.be/WHO6G67GJbM

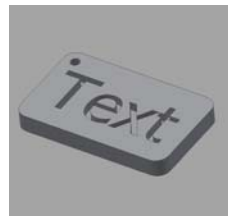
AM in Construction https://youtu.be/wCzS2FZoB-I

AM in Construction https://youtu.be/GUdnrtnjT5Q

Module

Computer Aided Design (CAD)





Introduction to CAD
Keychain Activity

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Newport News Shipbuilding Launches the Digital Shipyard

2

Newport News Shipbuilding launches the digital shipyard

https://www.youtube.com/watch?v=93za-vO_ffs

Acknowledgment: This work is supported by National Science Foundation grant 1749566

3

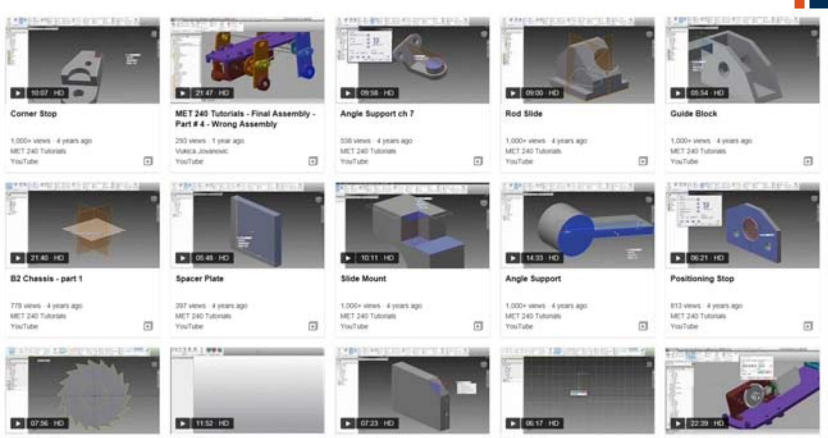
Digital twin: Making your asset smarter with the digital twin

Digital twin: Making your asset smarter with the digital twin

https://www.youtube.com/watch?v=8ger1wrAonM

Acknowledgment: This work is supported by National Science Foundation grant 1749566

More Autodesk Inventor Tutorials Search for "MET 240 Tutorials"



Acknowledgment: This work is supported by National Science Foundation grant 1749566

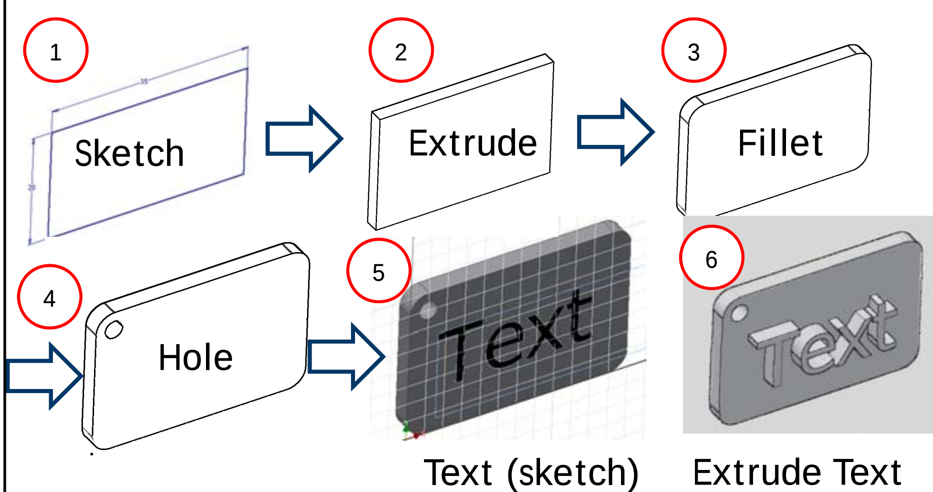
Objectives

- Create simple extruded solid model
- Understand feature interactions
- Create 2-D sketches
- Create and edit parametric dimensions
- Use the Part Browser

https://www.sdcpublications.com/Textbooks/Learning-Autodesk-Inventor-2018/ISBN/978-1-63057-131-3/

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Sketch > Extrude > Fillet > Hole > Text > Extrude Text

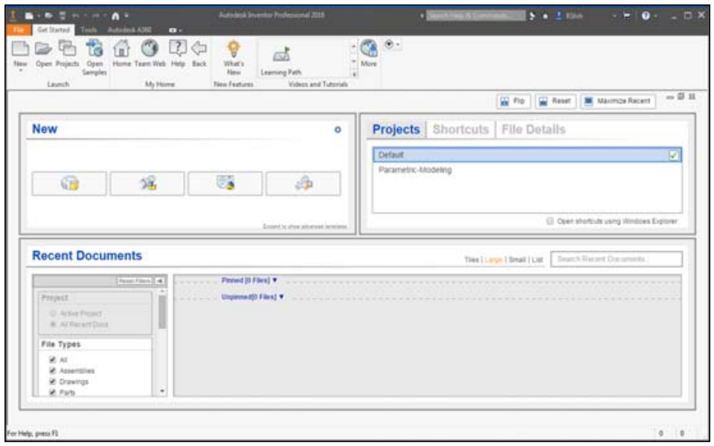


Acknowledgment: This work is supported by National Science Foundation grant 1749566

Getting Started

Autodesk Inventor Professional

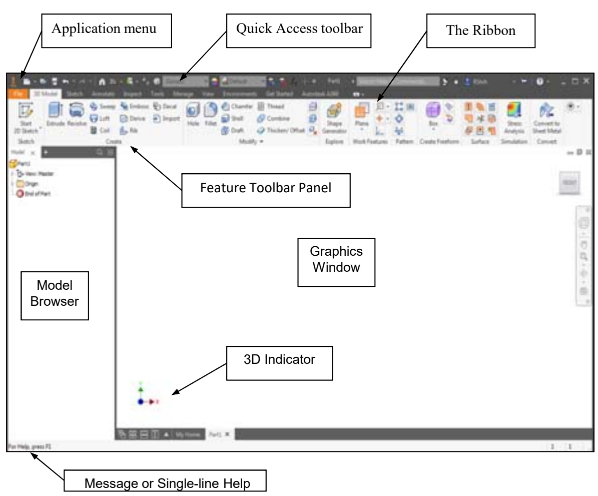
Start > Programs > Autodesk > Autodesk Inventor



https://www.sdcpublications.com/Textbooks/Learning-Autodesk-Inventor-2018/ISBN/978-1-63057-131-3/

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Autodesk Inventor Screen Layout

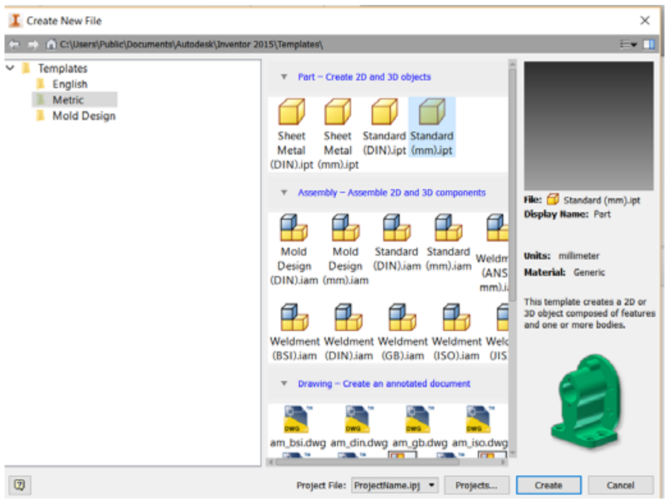


https://www.sdcpublications.com/Textbooks/Learning-Autodesk-Inventor-2018/ISBN/978-1-63057-131-3/

https://www.autodesk.com/products/inventor/overview

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Create > New File > Metric > Standard (mm).ipt



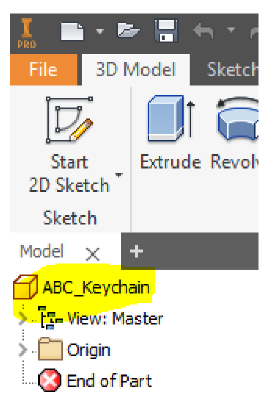
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Save File

- Save file with the following name to your USB:

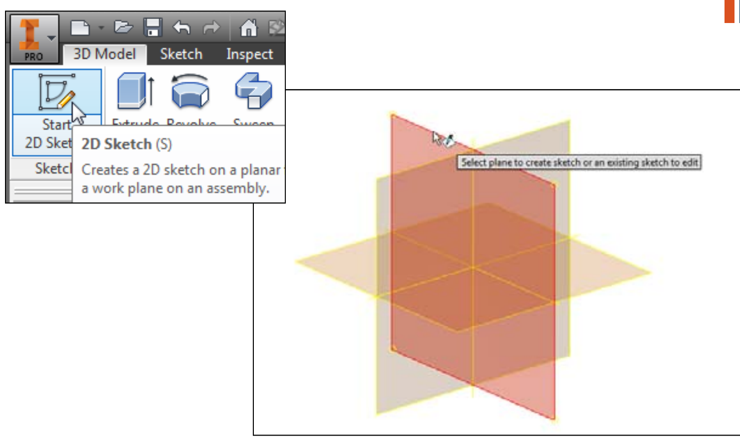
  Initials_Keychain.ipt

e.g.



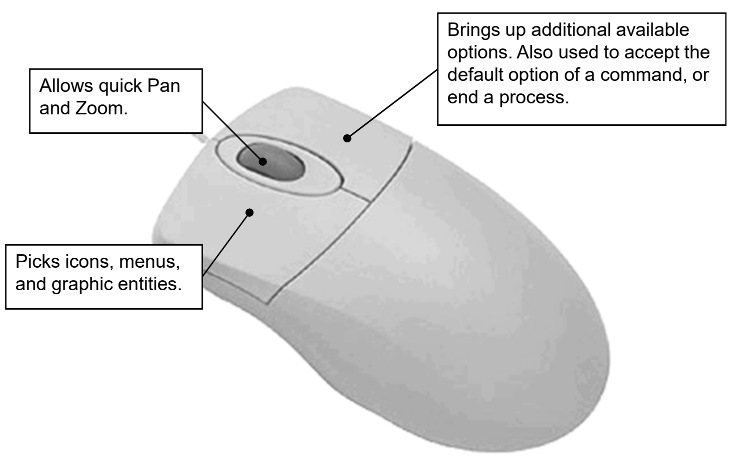
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Sketch Plane



Acknowledgment: This work is supported by National Science Foundation grant 1749566

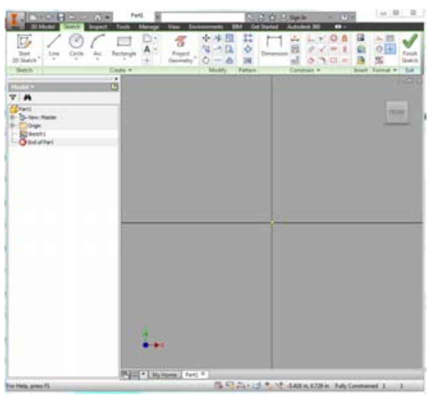
Mouse Buttons

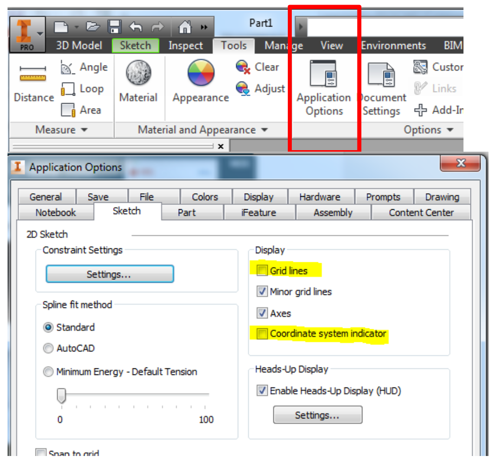


https://www.sdcpublications.com/Textbooks/Learning-Autodesk-Inventor-2018/ISBN/978-1-63057-131-3/

Acknowledgment: This work is supported by National Science Foundation grant 1749566

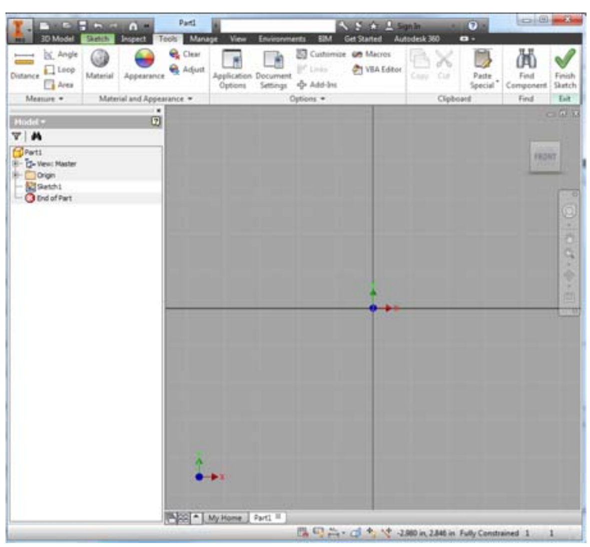
Tools > Application Options > Sketch > Turn On Grid lines & Coordinate System Indicator





Acknowledgment: This work is supported by National Science Foundation grant 1749566

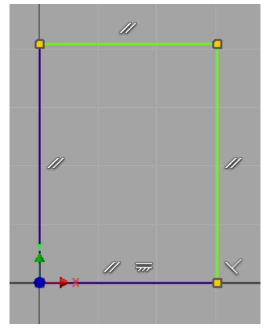
UCS and Grid are active – see below



Acknowledgment: This work is supported by National Science Foundation grant 1749566

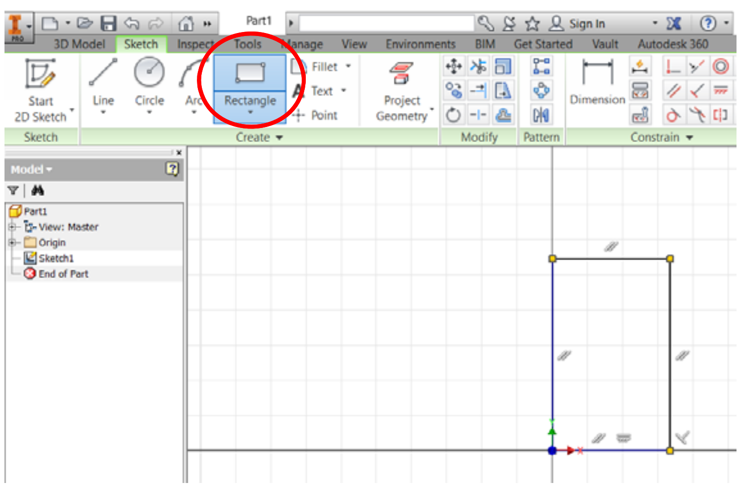
Creating Rough Sketches

- As the name implies, a rough sketch is not precise at all.
- When sketching, we simply sketch the geometry so that it closely resembles the desired shape.
- Precise scale or lengths are not needed.
- Create a sketch that is proportional to the desired shape. Concentrate on the shapes and forms of the design.



Acknowledgment: This work is supported by National Science Foundation grant 1749566

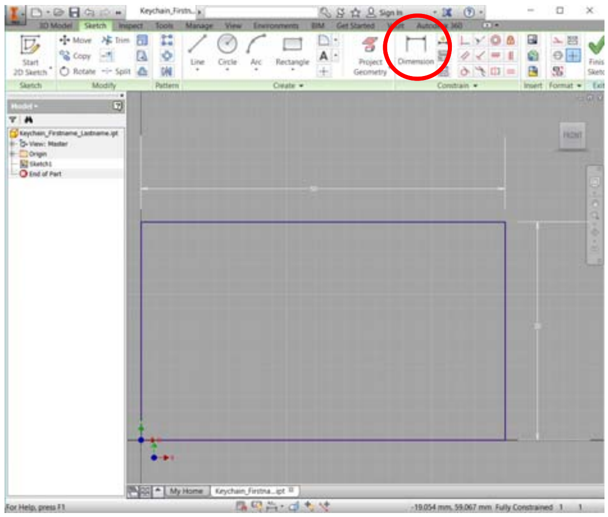
Draw the Rectangle



Use USC (User coordinate system) to avoid unnecessary dimensions

Acknowledgment: This work is supported by National Science Foundation grant 1749566

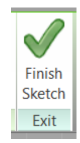
Add Dimensions



e.g. 50 mm x 30 mm

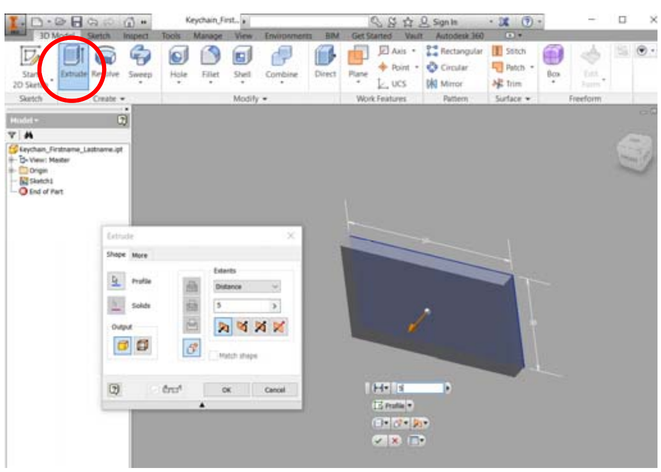
Select Dimension >
Click on a line>
Click outside where you want number to appear

Finish Sketch



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Extrude



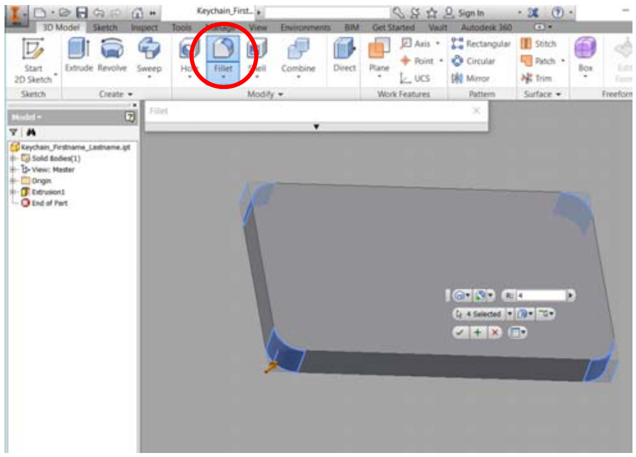
Hold Shift + press the middle mouse key > 3D orbit

Extrude >
Select Rectangle (Profile) >
Extents > Distance >
5 mm

View > Visual Style >
Shaded with Edges

Acknowledgment: This work is supported by National Science Foundation grant 1749566

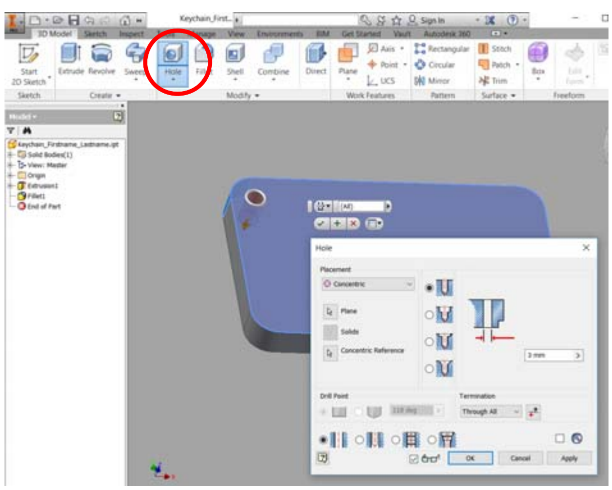
Fillet



3D Model >
Fillet >
Select four lines
in the corners >
4 mm

Acknowledgment: This work is supported by National Science Foundation grant 1749566

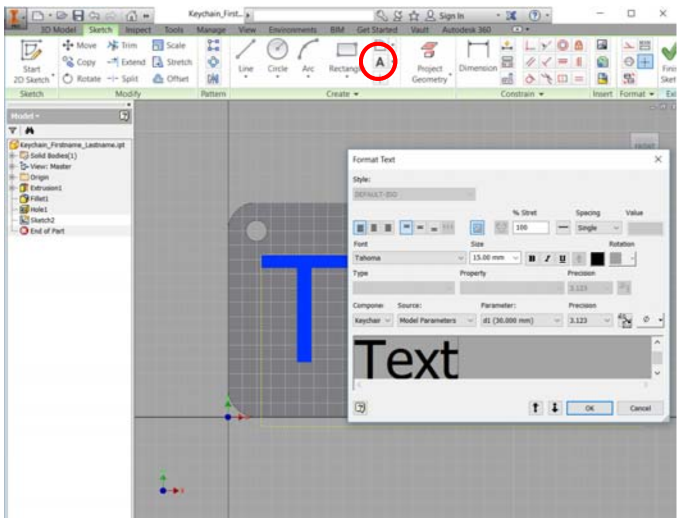
Hole (Placement - Concentric, Termination - Through All)



3D Model >
Hole > Placement >
Concentric >
Select a plane on
which you are
drilling a hole >
Select a concentric
reference (a fillet)
to match their axis >
Termination >
Through All >
Diameter > 3 mm

Acknowledgment: This work is supported by National Science Foundation grant 1749566

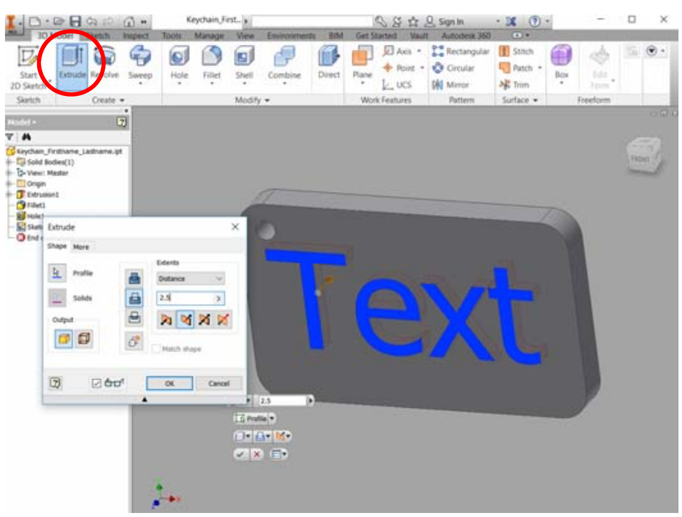
Text



Sketch >
Select a plane >
Create >
Text >
Select a location >
Change size to 15 mm >
Write text that is to be engraved in a keychain >
OK >
Left click and hold on a text to adjust the position >
Finish sketch

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Extrude > Text

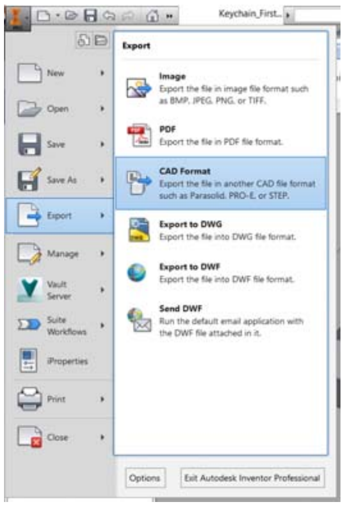


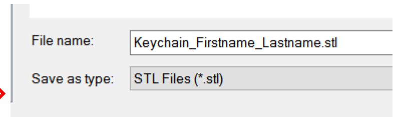
3D Model >
Extrude >
Select text >
Change to Cut >

Extents >
Distance>
2.5 mm >
OK

Acknowledgment: This work is supported by National Science Foundation grant 1749566

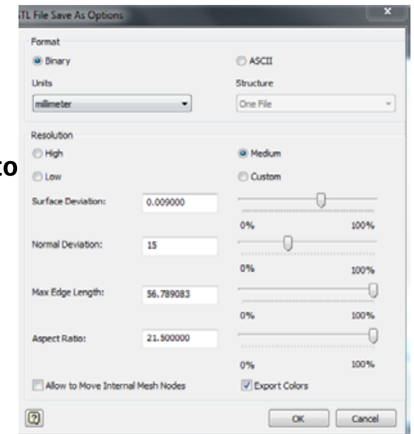
Export > CAD Format > STL





Save file to the SD Card

You will learn how to create a G code needed for the printer in the following module



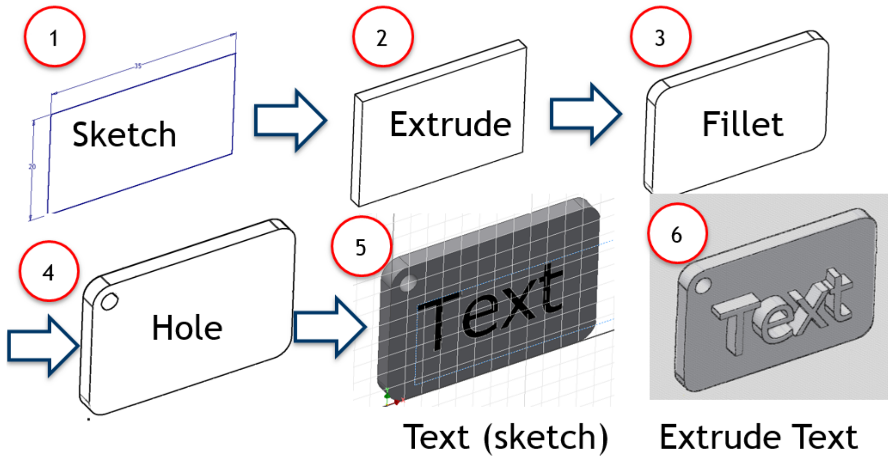
Acknowledgment: This work is supported by National Science Foundation grant 1749566

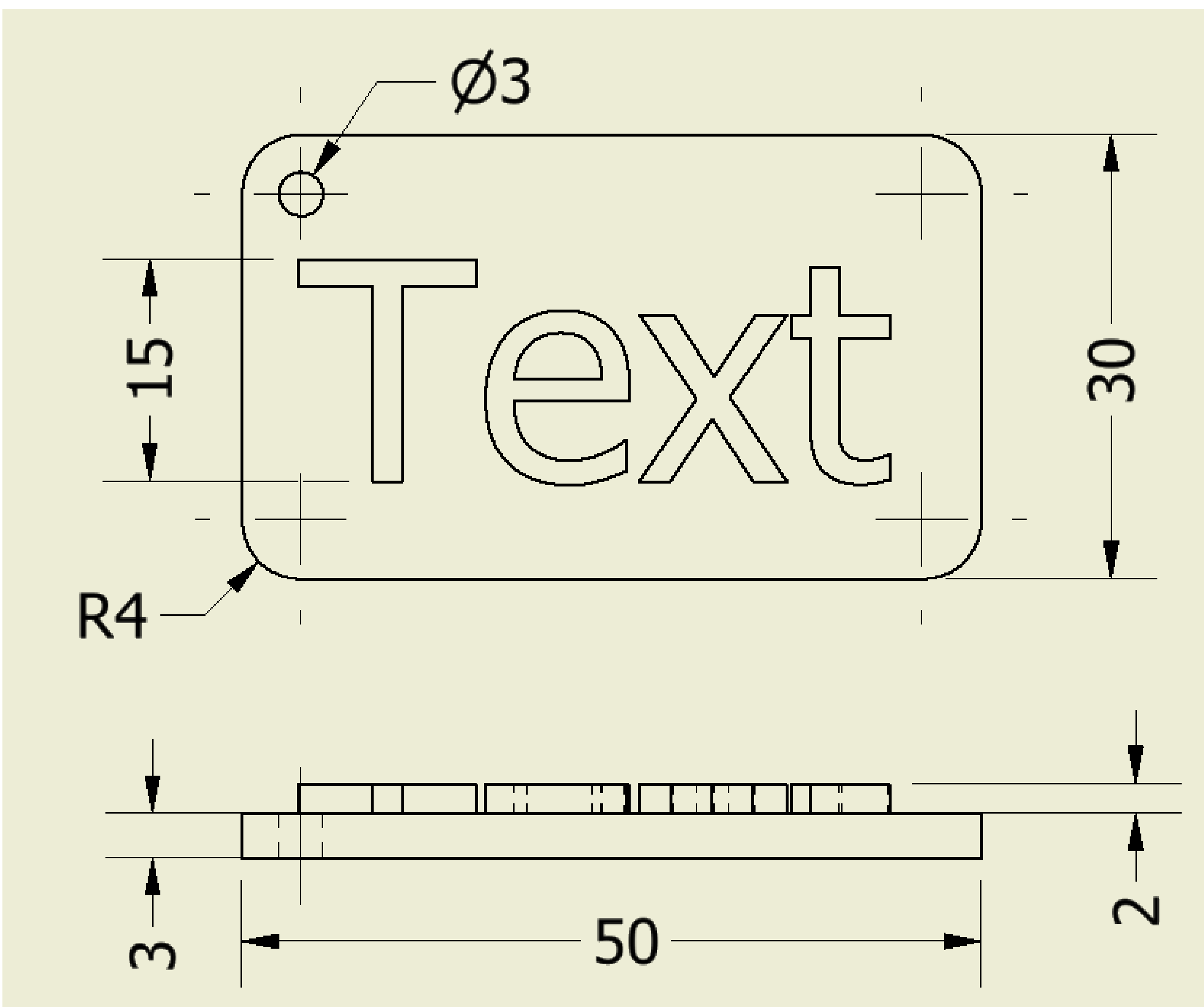
References

- Autodesk (2018) Inventor Professional 2018 – CAD Software https://www.autodesk.com/products/inventor/overview
- Shih, R.H. (2017) Learning Autodesk Inventor 2018 - Modeling, Assembly and Analysis, ISBN: 978-1-63057-131-3, SDC Publications https://www.sdcpublications.com/Textbooks/Learning-Autodesk-Inventor-2018/ISBN/978-1-63057-131-3/
- Jovanovic, Vukica (2017) Keychain Design https://www.youtube.com/watch?v=s-Yud6-DbNc

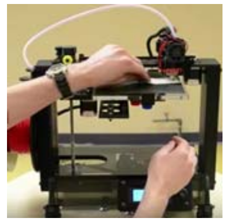
Acknowledgment: This work is supported by National Science Foundation grant 1749566

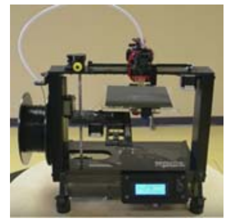
Keychain


1
Sketch
2
Extrude
3
Fillet
4
Hole
5
Text
Text (sketch)
6
Text
Extrude Text


Ø3
Text
15
30
R4
3
50
2

CAD to G Code





Module Content:

- Slicing – from CAD to G code

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Objectives

- Increase awareness
  - 3D Printing
  - 3D Printing Technologies
- Learn
  - How to convert a 3D design into G code
  - Slic3r software
  - Send a part to be 3D printed

Technological Literacy

ACTIVITY

Introduction of Participants

- Name
- Are you familiar with 3D printing? How?

AM Technologies You Will Learn

Photopolymer Jetting and Fused Deposition Modeling

Additive Manufacturing Technologies You Will Use

3D Printing Technologies You Will Use

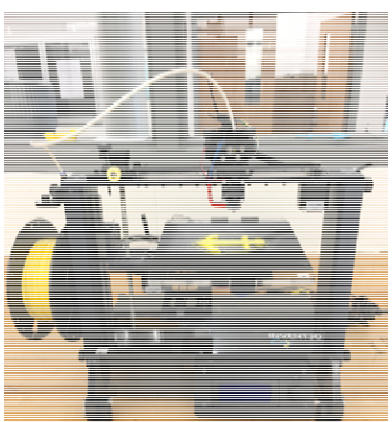
- Object 30 Prime
  - Materials – 8 options: four rigid opaque, transparent, high-temperature, and two simulated polypropylene options
  - Building Envelope – 11.8 x 7.8 x 5.9 in
  - Layer thickness – 0.0011in – 0.0006in 0.028mm – 0.016mm (transparent material)

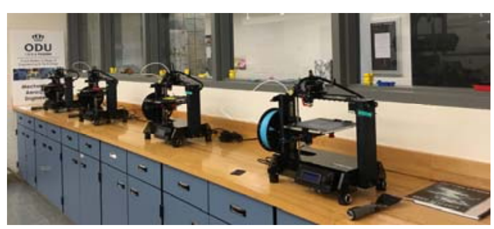
NSF Grant # 1749566
https://sites.wp.odu.edu/oduvetmaker/

Object 30 Prime (Photopolymer Jetting - PJ)

3D PRINTER SPECIFICATIONS	
Model Materials	Rigid Opaque (VeroWhitePlus™, VeroGray™, VeroBlue™, VeroBlackPlus™) Transparent (RGD720 and VeroClear™) Simulated Polypropylene (Rigur™ and Durus™) High Temperature Rubber-like (TangoGray™ and TangoBlack™) Bio-compatible
Support Material	SUP705 (WaterJet removable) SUP706 (soluble)
Maximum Build Size (XYZ)	294 x 192 x 148.6 mm (11.57 x 7.55 x 5.85 in.)
System Size and Weight	82.5 × 62 × 59 cm (32.28 × 24.4 × 23.22 in.); 106 kg (234 lbs)
Resolution	**X-axis:** 600 dpi; **Y-axis:** 600 dpi; **Z-axis:** 1600 dpi
Accuracy	0.1 mm (0.0039 in.) varies depending on part geometry, size, orientation, material and post-processing method
Minimum Layer Thickness	28 microns (0.0011 in.) for Tango materials; 16 microns (0.0006 in.) for all other materials
Build Modes	Draft (36 micron); High Speed (28 micron); High Quality (16 micron)
Software	Objet Studio™ intuitive 3D printing software
Workstation Compatibility	Windows XP/Windows 7/Windows 8
Network Connectivity	Ethernet TCP/IP 10/100 base T
Operating Conditions	Temperature 18-25°C (64-77°F); relative humidity 30-70%

3D Printing Technologies You Will Use





Invent 3D Printer
(Fused Deposition Modeling - FDM)

- Building Envelope – 8 x 8 x 12 in
- Layer thickness – 0.007 to 0.010 in

NSF Grant # 1749566
https://sites.wp.odu.edu/oduvetmaker/

Convert from .STL to .G

3D model

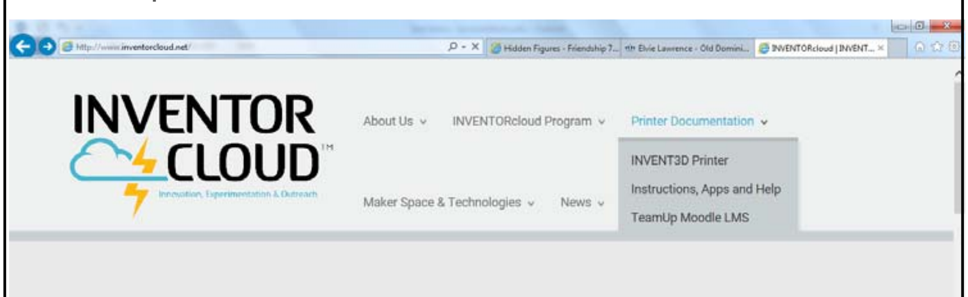
2D slices

Layered Material Deposition Instructions

.STL → .G

Slic3r – Layer Preparation Software

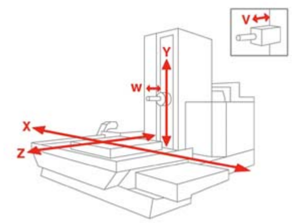
- 3D Model input formats: STL, OBJ, AMF
- Output software: .G



http://www.inventorcloud.net/installers/

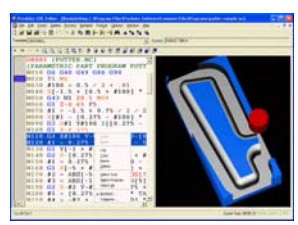
G Code

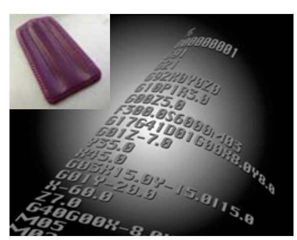
- The most widely used numerical control (NC) programming language
  - G-code is a language in which people tell computerized machine tools how to make something.
  - The "how" is defined by instructions on where to move, how fast to move, and what path to move.
- It is used mainly in computer-aided manufacturing to control automated machine tools.



G Code – CNC & AM

- In a CNC, the cutting tool is moved according to the instructions from the G-code through a toolpath and cuts away material to leave only the finished workpiece
- The same concept also extends to additive manufacturing, with the G-code providing preparatory and in-process commands, controlling the type of motion to extrude material, and the extrusion of material:
  - rapid positioning
  - linear feed
  - circular feed
  - what offset value to use
  - X, Y, Z - Absolute or incremental position





Invent3D Printer

- Needs G code to print a 3D model
- Uses an SD card
  - G-code is transferred to printer with the SD card

Slic3r
G-code generator for 3D printers

Slic3r Software

- Download from:

http://www.inventorcloud.net/installers/

- Install:

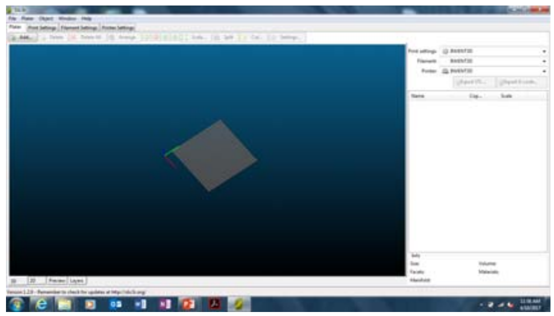
Slic3r (new heads) for Windows (1.2.9 v3s8)

Slic3r
G-code generator for 3D printers

15

Using Slic3r

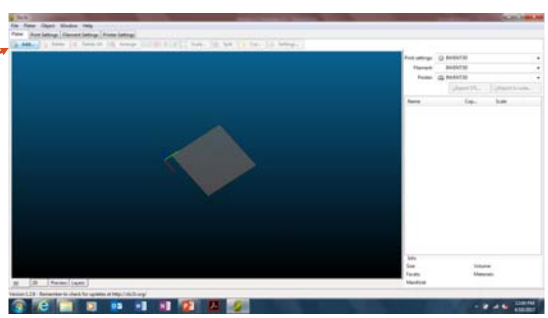
- Open Slic3r



16

Using Slic3r

- In the *Platter tab* (default) – go to Add (top left, under the *Platter tab*)
- Choose a .STL, .OBJ, or AMF file from your computer. The 3D part will then appear on the platter.

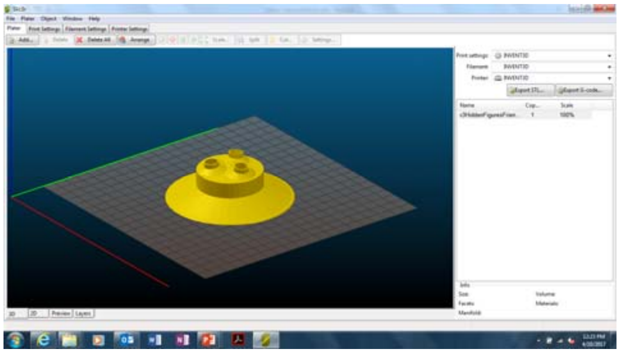


NSF Grant # 1749566

17

Using Slic3r

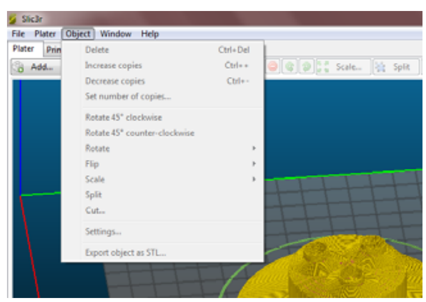
- The part (or object) is loaded into the platter centered and, most likely, in the best orientation.



18

Using Slic3r

- Under the object menu:
  - The part can be removed from the platter
  - More copies of the part can be added, or later deleted
  - The part can be rotated, flipped, scaled, cut, or split



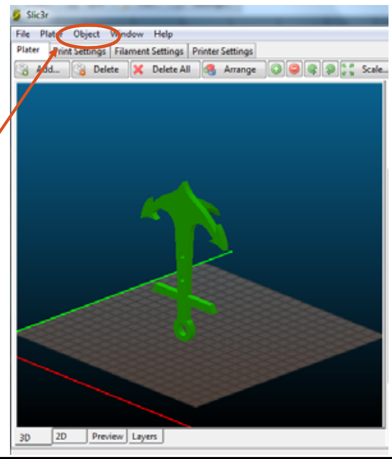
19

Using Slic3r

- The part (or object) can be rotated to the user’s best orientation.

1. The part should be selected (green)

2. On the **Object** tab, rotate the part as needed (this will depend on the axis how the part was designed or read by Slic3r)

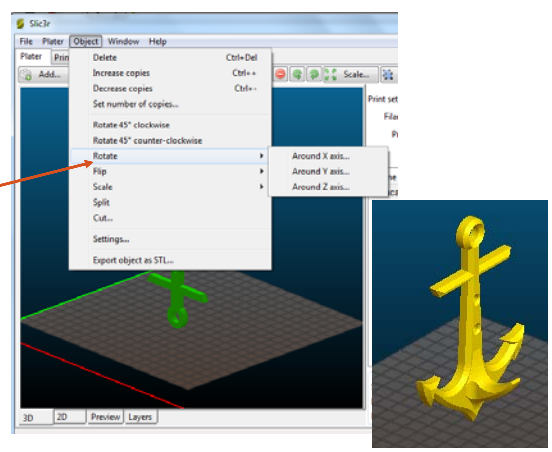


20

Using Slic3r

- The part (or object) can be rotated to the user’s best orientation.

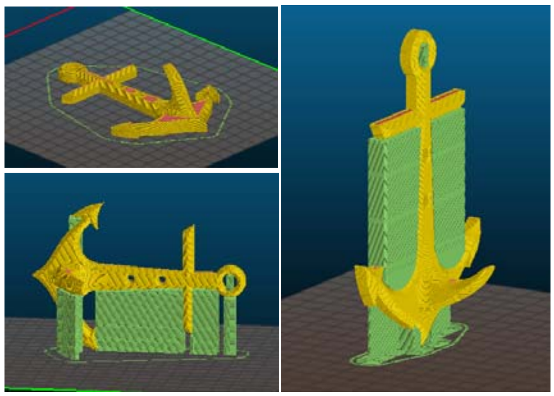
3. On the **Object** tab, rotate or flip the part as needed (this will depend on the axis how the part was designed or read by Slic3r)



21

Using Slic3r

- The part (or object) can be rotated to the user's best orientation.



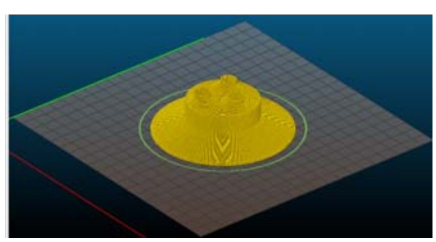
The **load, force, or stress** that it will support, the **fit** of external features (shrinkage) versus the layered orientation.

Support material.
Printing time.
Others??

22

Using Slic3r

- Depending on how the part was designed (inches or mm), sometimes parts must be scaled to be printed with the right dimensions.
- The platter size is set at 200 x 200 – this is a default value (do NOT change)
  - Printer Settings --> General --> Bed size
  - This is in mm or ~ 8 x 8 inches
- When scaling, note that each square in the plater is 10 x 10 mm or approximately ½ inch (0.4 x 0.4in)

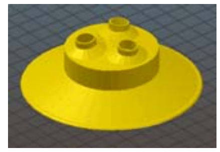


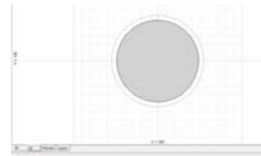
mm → 1000%   in → 2400%

NSF Grant # 1749566
https://sites.wp.odu.edu/oduvetmaker/

23

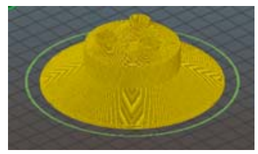
Using Slic3r





- In the four tabs a the bottom left, the part can be seen from different perspectives:

  - 3D (default – close to a solid part)
  - 2D (top view – not much information, area it occupies and position in platter)
  - Preview (layered view)
  - Layers (each individual layer)

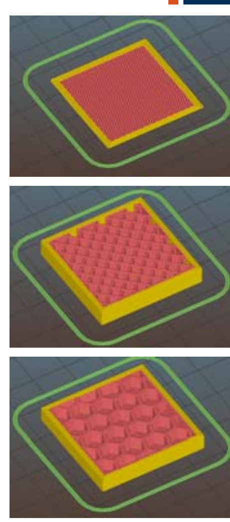


24

Using Slic3r - Terminology

- Shells - perimeters or outlines
  - Vertical – Default: 3 – outline of the part
  - Horizontal – Default: 5 – bottom and top layers which are created with a “solid” or very close infill
- Infill – center section of a part
  - Density – Default: 20%
  - Pattern – Default: Rectilinear 45º

    Other options: honeycomb, concentric, spiral, etc.



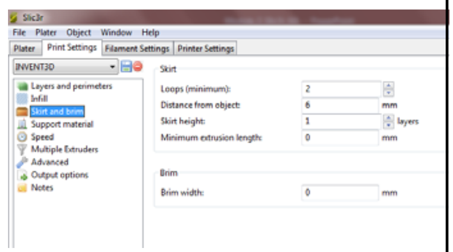
25

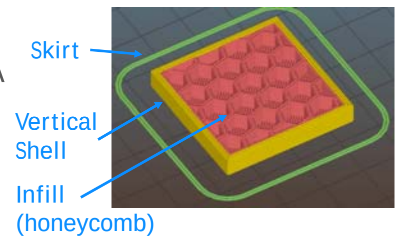
Using Slic3r - Terminology

- Skirt – outline where the part to be printed fits

  DO NOT CHANGE – use to check calibration of printer

- Brim – edge attached to the part to increase area attached to platform
  - USE WHEN PRINTING PARTS WITH A SMALL SURFACE AREA avoid detachment from platform during fabrication
  - USE to avoid warping

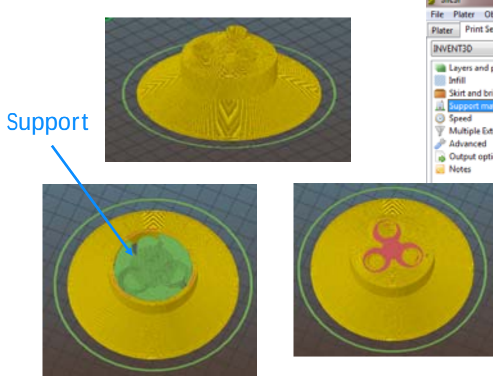


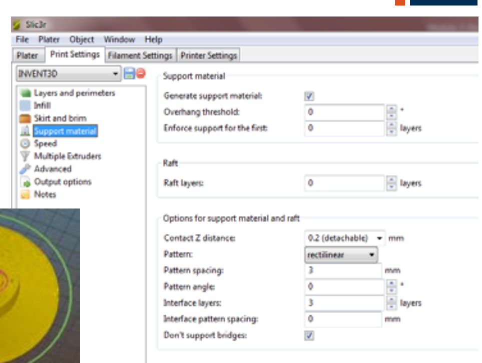


26

Using Slic3r - Terminology

- Support Material – used in parts with overhanging or bridged features



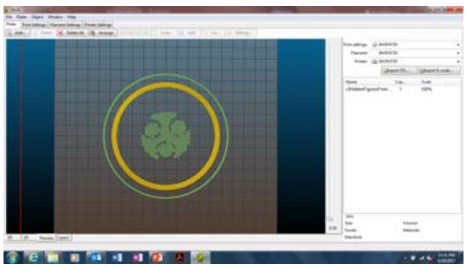


- Other parameters – do not change, leave as default

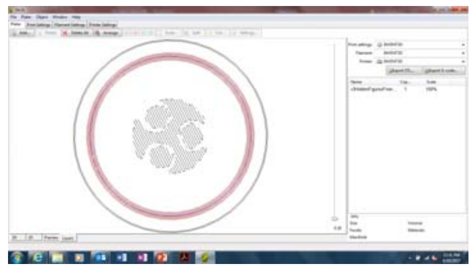
Slic3r – Going through the layers

27

Top view in the preview tab – layers underneath are noticeable



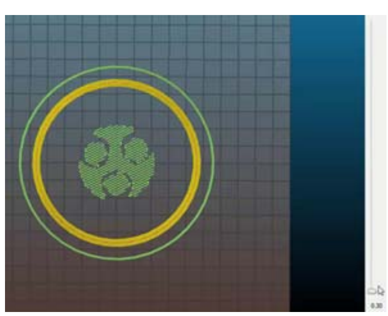
Layers tab – each individual layer can be seen (path the extruding head will follow to create the part)



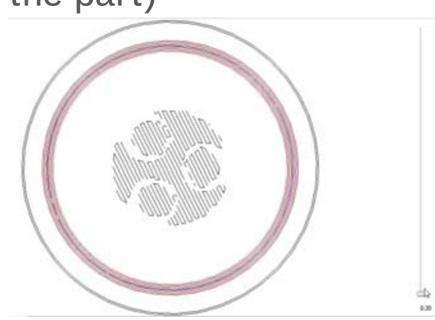
Slic3r – Going through the layers

28

Top view in the preview tab – layers underneath are noticeable



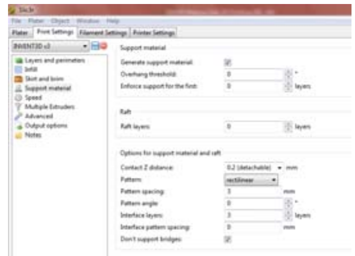
Layers tab – each individual layer can be seen (path the extruding head will follow to create the part)

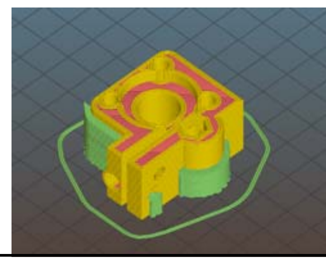


29

Slic3r – Modifying Support Structures

- On occasion, it would be best to modify the support structures to improve post-processing
  - Save time
- Trial-and-error process
  - IMPORTANT – to inspect the layers prior to print





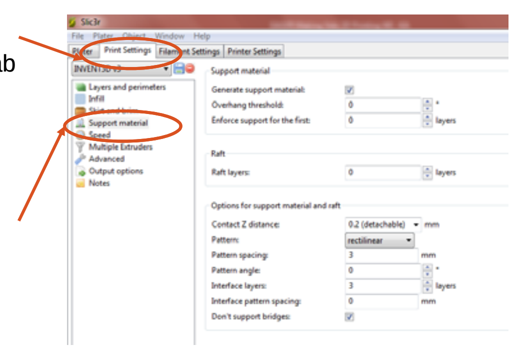
30

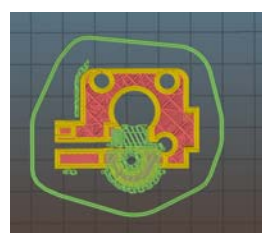
Slic3r – Modifying Support Structures

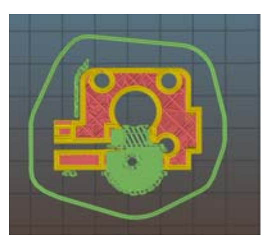
Default Settings

1. On the **Print Settings** tab

2. Select Support Material







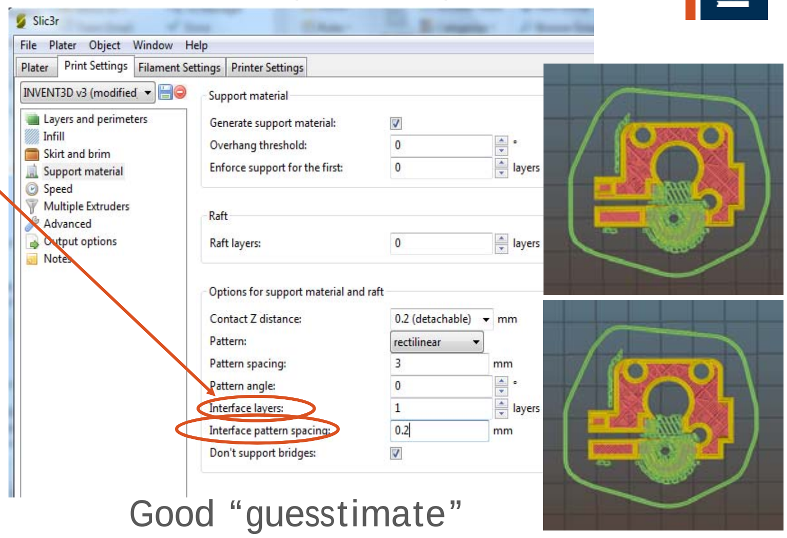
31

Slic3r – Modifying Support Structures

Change Settings

3. On the **Interface Layers** use 1, so there is only one “solid” layer

4. On the **Interface Patten Spacing** use 0.2mm

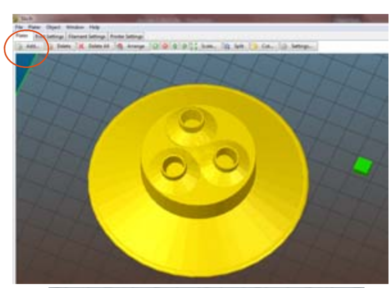


Good “guesstimate”

32

Slic3r – Sending Several Parts to Print at the Same Time

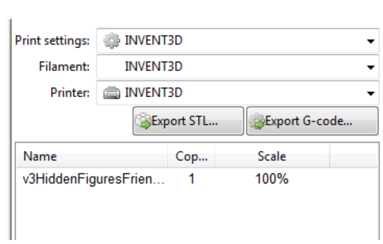
- On occasion, it would be best to have several different parts printed at the same time
  - If several parts fit on the platform, this can be added using the Add command (top left)
  - If the part need to be scaled, rotated, or increase copies, make sure it is selected (green color, mouse click)
- Then it is saved to be transferred to the printer



33

Slic3r – Exporting the .G code

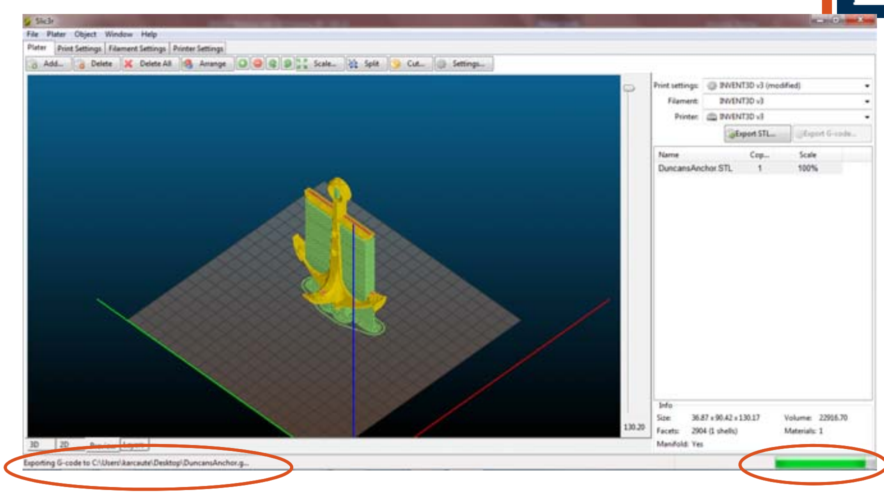
- Once the desired settings have been changed, the layered part is exported as .G code to be printed
  - On the top right, click on Export G-code
- Save on your computer and/or SD card to be transferred to the printer



34

Slic3r – Exporting the .G code

- Make sure the G code was exported



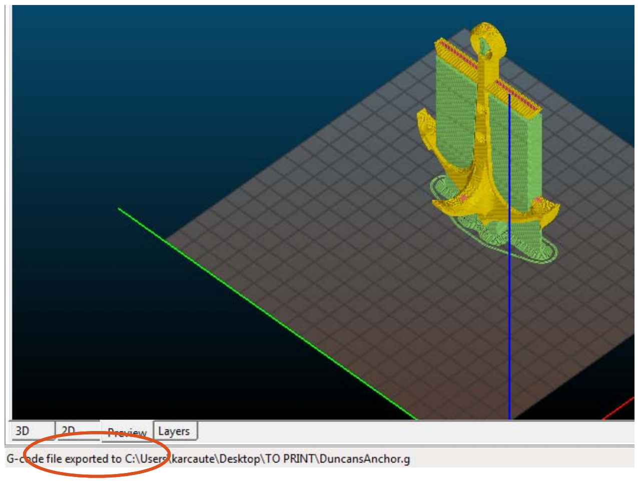
In progress - Exporting

Completion Bar

35

Slic3r – Exporting the .G code

- Make sure the G code was exported



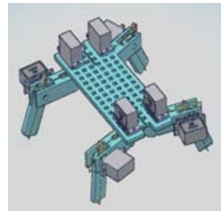
Exported directory

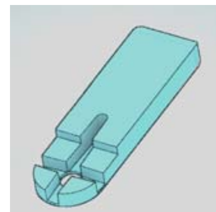
Sending a Design to be Printed

1. Make sure you have the G code of the design in the SD card and the printer is leveled and ready to print
   - Printer will be leveled
2. Send the part to print
   - Select PREPARE > Print from SD card > FILENAME
     - The temperature reading will indicate the temperature in the print head (warming up) and the temperature for extruding (220ºC)
     - The fan will turn on at ~50ºC (if not, cancel print job) → WARNING
     - Once the temperature stabilizes at 220ºC the printer will go to AUTOHOME, and extrude some material before starting the print
3. Be ready to check the skirt (outline) to make sure it is continuous and well attached to the platform

Module

Computer Aided Design (CAD)





Parametric Modeling Fundamentals

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Objectives

- Introduce participants into Feature-Based Parametric Modeling design methodology
- Setup correct units and sketch options
- Review user interface & mouse buttons
- Understand Autodesk Inventor screen layout

https://www.sdcpublications.com/Textbooks/Learning-Autodesk-Inventor-2018/ISBN/978-1-63057-131-3/

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Introduction

- **Parametric**
  - Geometric definitions of designs can be varied at any time in the design process:
    - E.g. dimensions
- **Parametric modeling**
  - identifying and creating the key features of the design with the aid of computer software.
- **Feature-based** parametric modeling
  - Build a part from features
    - E.g. Extrusion, Hole, Fillet, etc.

Acknowledgment: This work is supported by National Science Foundation grant 1749566

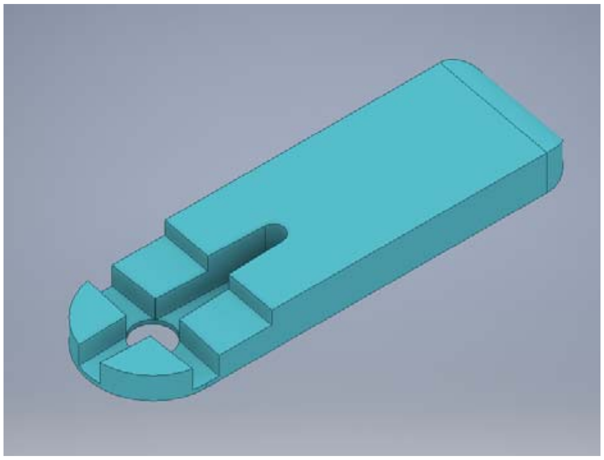
Feature-Based Parametric Modeling

Parametric part modeling process steps:

- Create a rough 2D sketch for the base feature
- Apply/modify constraints and dimensions
- Extrude, revolve, or sweep 2D sketch to create the base solid feature
- Add additional parametric features by identifying feature relations
- Perform analyses and refine the design
- Create the desired drawing to document the design

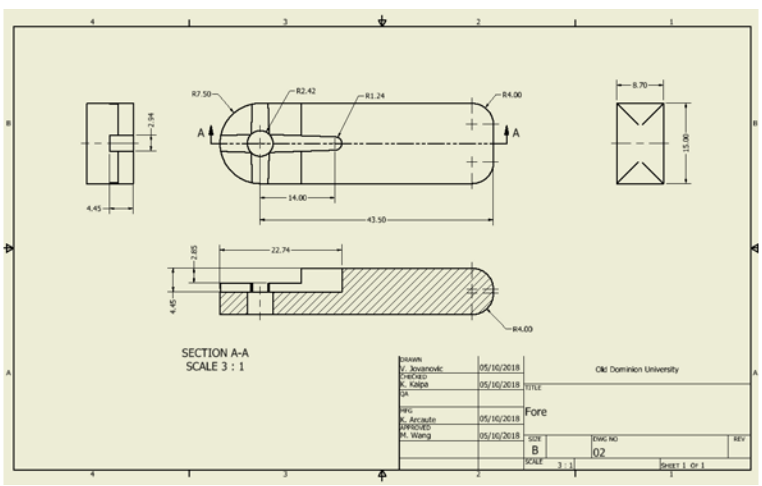
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Lizard Leg Design



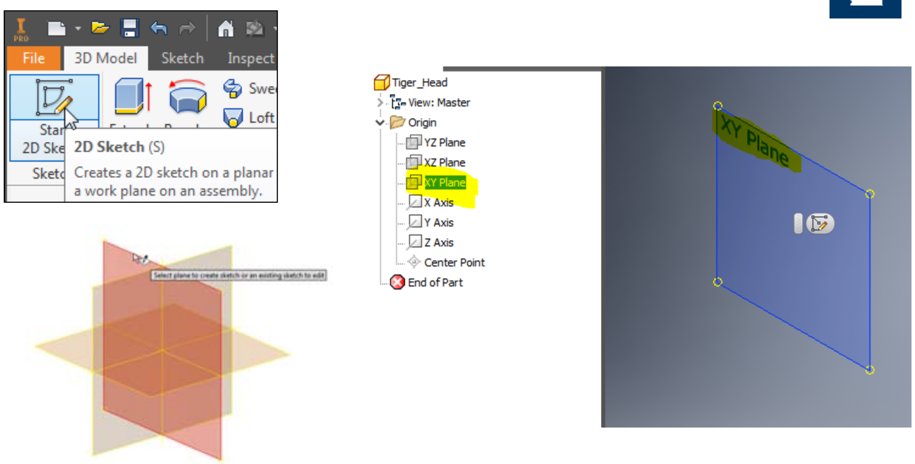
Acknowledgment: This work is supported by National Science Foundation grant 1749566

2D Engineering Drawing



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Sketch Plane - Select XY Plane



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Creating Rough Sketches

- Simply sketch the geometry so that it closely resembles the desired shape
- Precise scale or lengths are not needed
- Dimension the first line so it is easier to scale later
- Sketches will be finalized by adding
  - Dimensions
  - Constraints

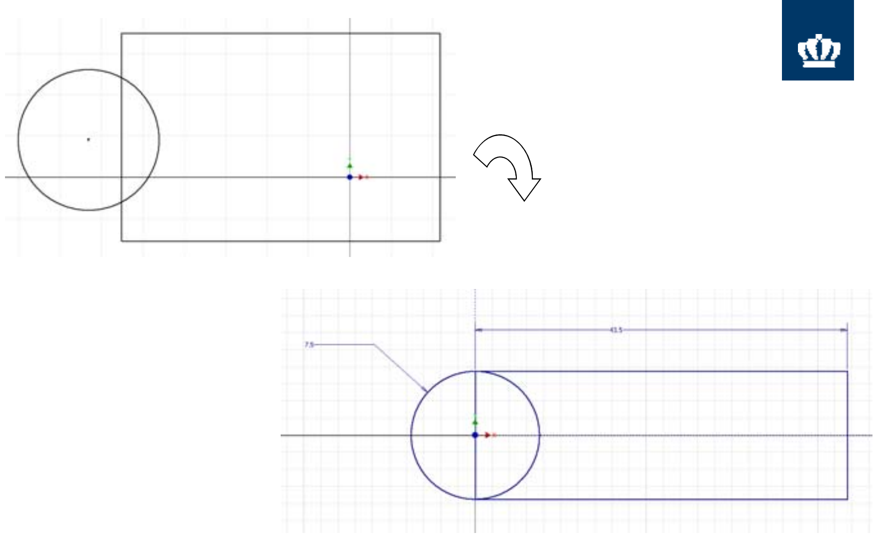
Acknowledgment: This work is supported by National Science Foundation grant 1749566

General Guidelines for Creating Sketches

- Create a sketch that is proportional to the desired shape
- Concentrate on the shapes and forms of the design
- Keep the sketches simple
- Leave out small geometry features
- Exaggerate the geometric features of the desired shape
- Draw the geometry so that it does not overlap
- Form a closed region so that it can be extruded later

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Creating Rough Sketches



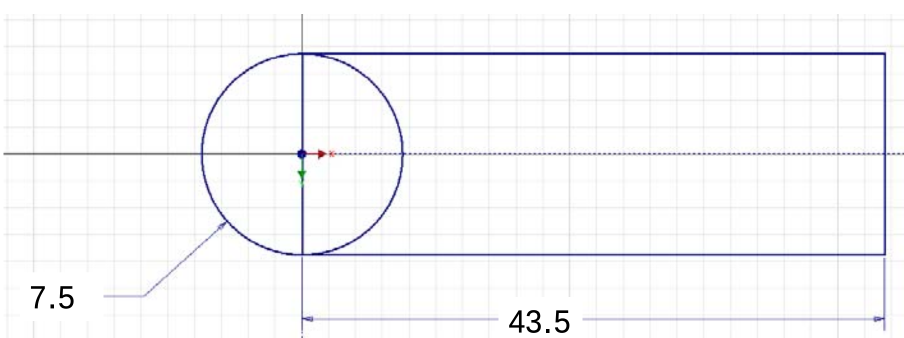
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Geometric Constraint Symbols

Vertical	indicates a line is vertical
Horizontal	indicates a line is horizontal
Dashed line	indicates the alignment is to the center point or endpoint of an entity
Parallel	indicates a line is parallel to other entities
Perpendicular	indicates a line is perpendicular to other entities
Coincident	indicates the cursor is at the endpoint of an entity
Concentric	indicates the cursor is at the center of an entity
Tangent	indicates the cursor is at tangency points to curves

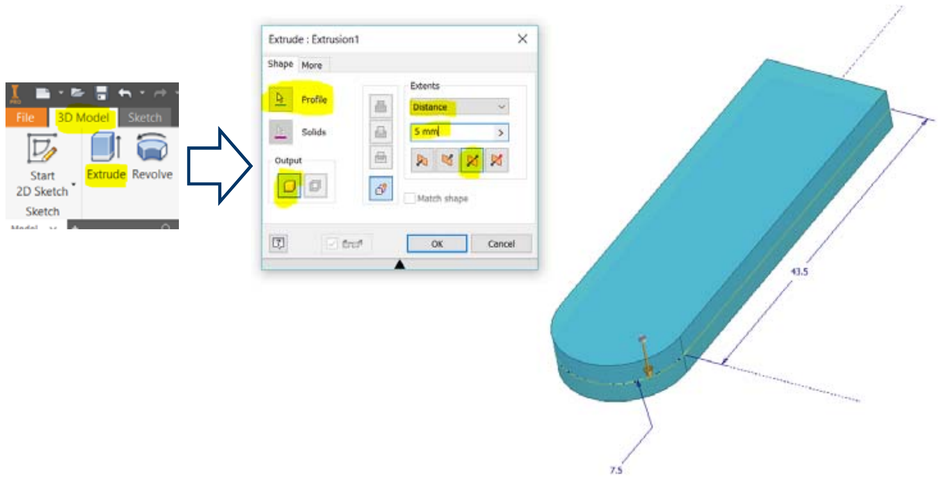
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Add Dimensions and Constraints



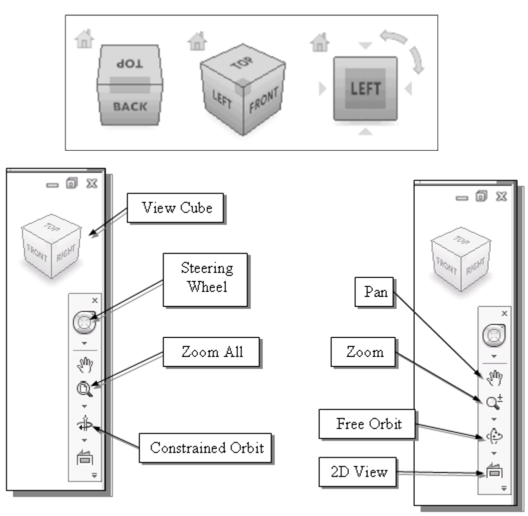
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Create a Solid Feature - Extrude



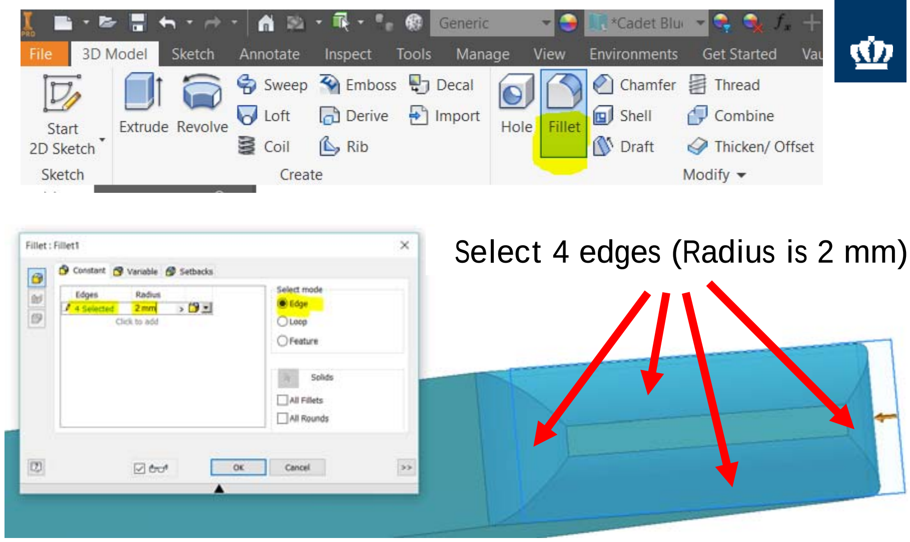
Acknowledgment.

Dynamic Viewing Functions



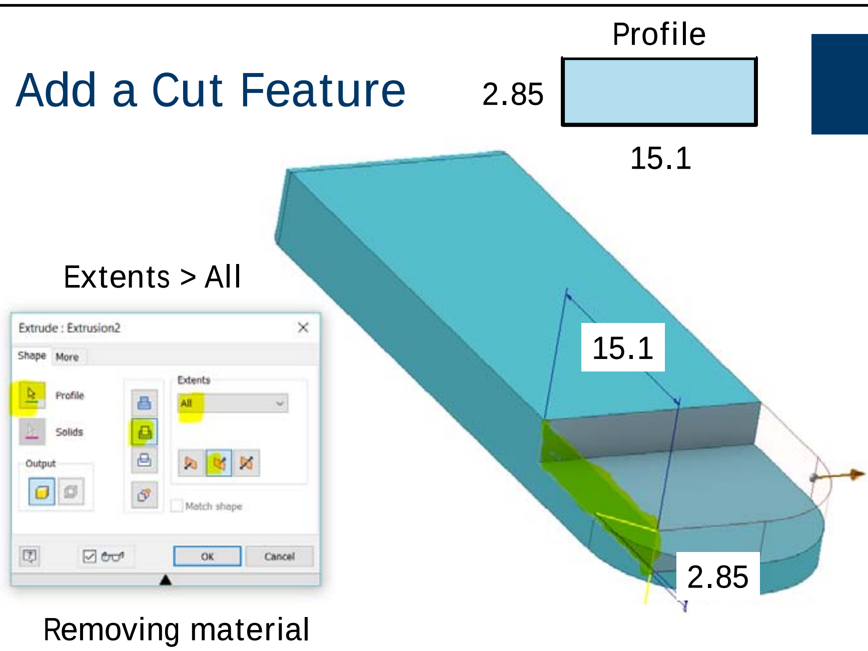
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Add Fillets



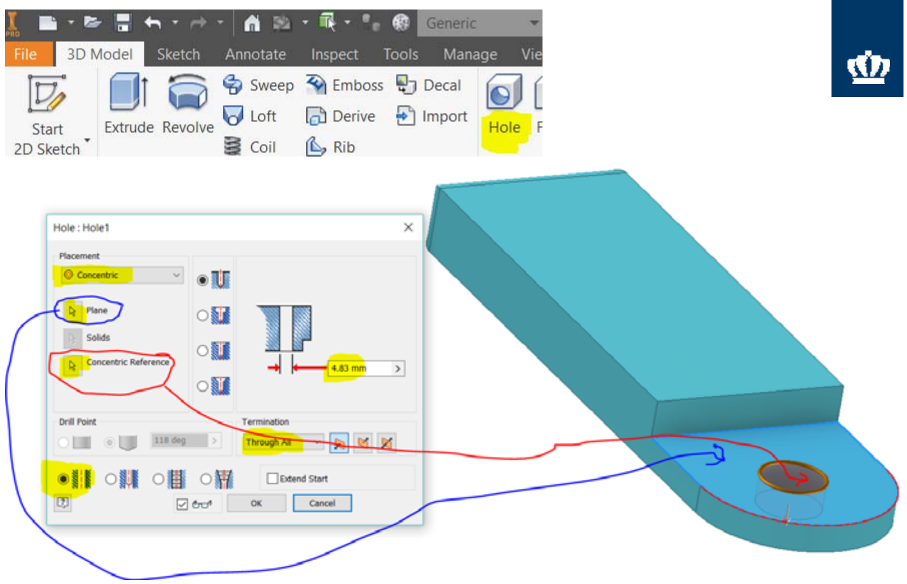
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Add a Cut Feature



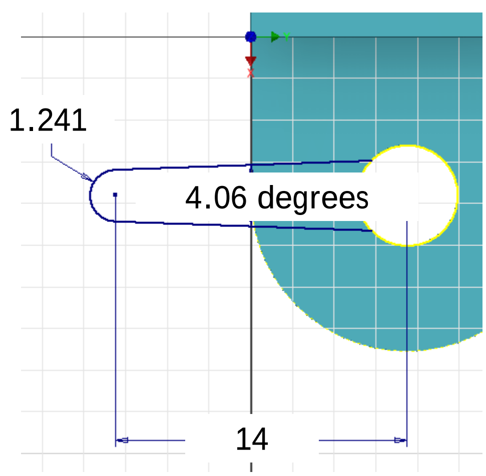
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Add a Hole Feature



Acknowledgment: This work is supported by National Science Foundation grant 1749566

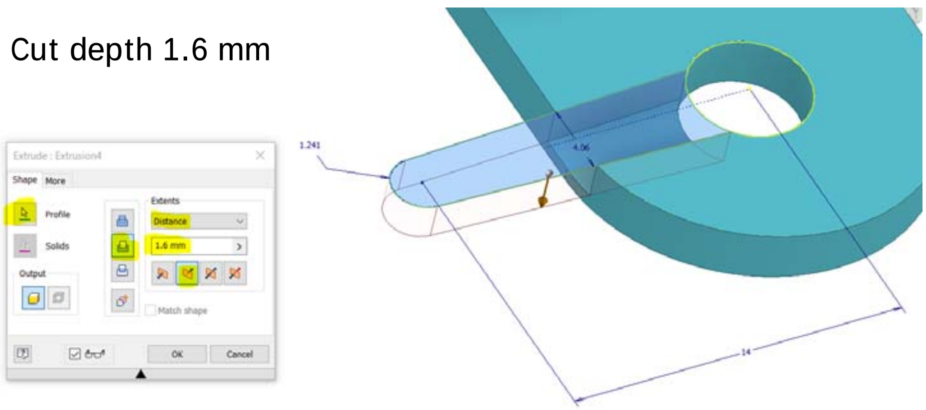
Add Another Extruded Feature



Acknowledgment: This work is supported by National Science Foundation grant 1749566

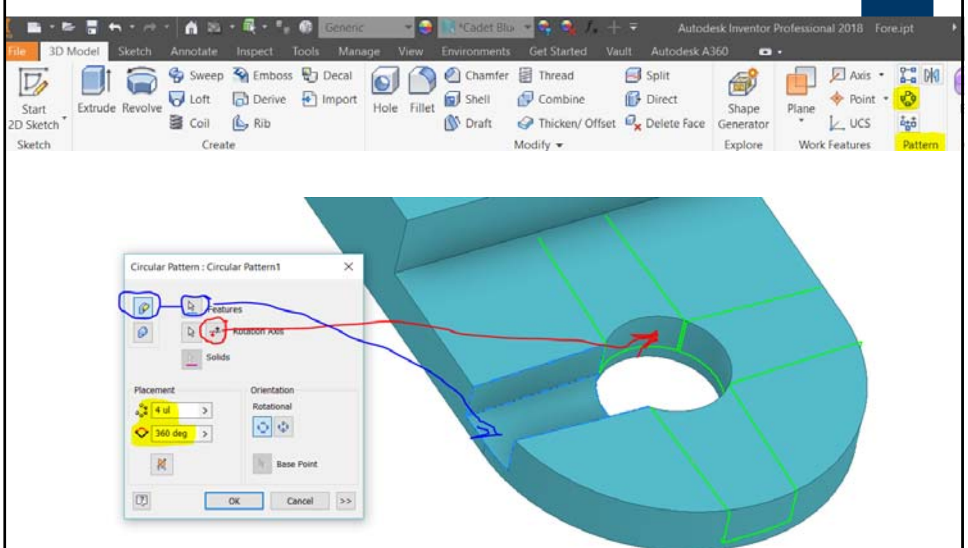
Feature : Extrude > Remove (Cut)

Cut depth 1.6 mm



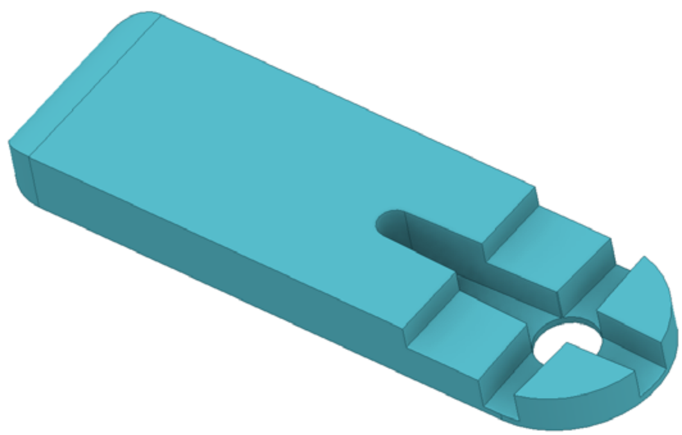
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Feature > Circular Pattern



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Finished Parametric Part

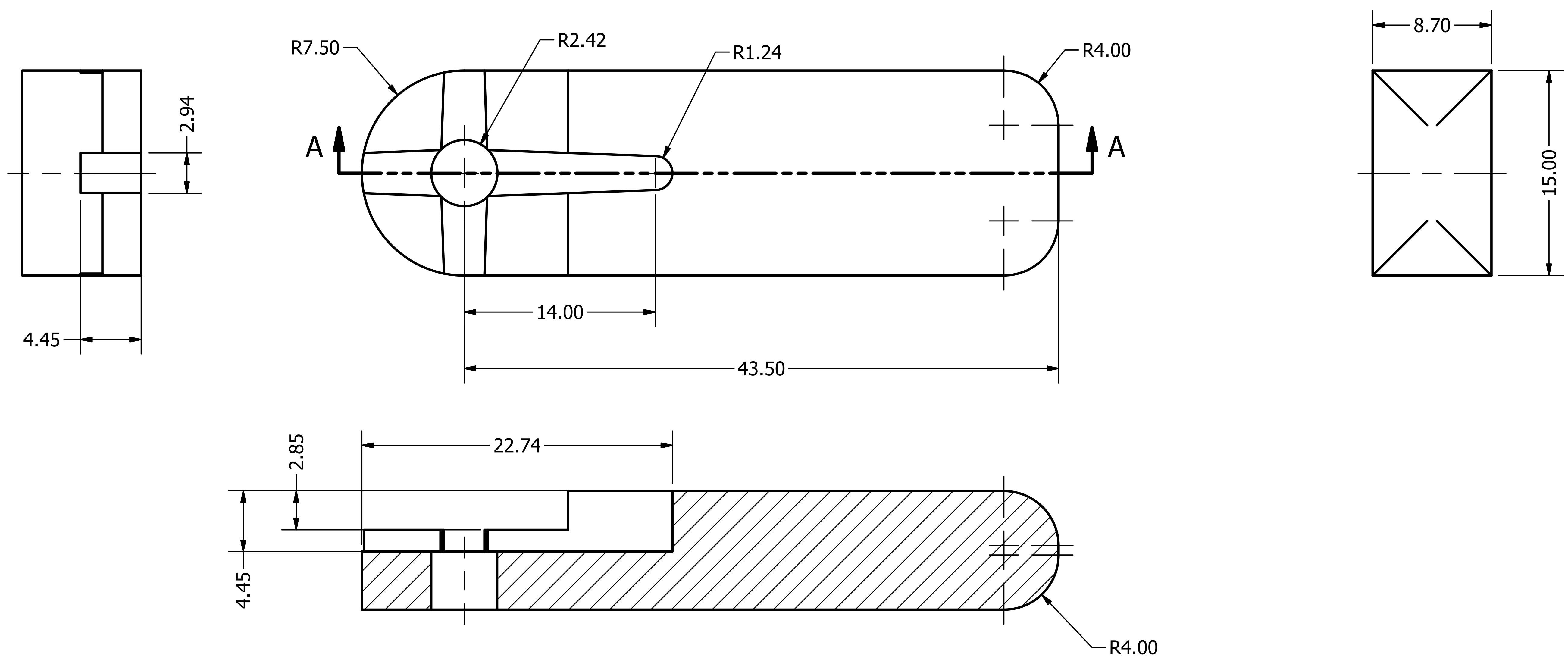


Acknowledgment: This work is supported by National Science Foundation grant 1749566

References

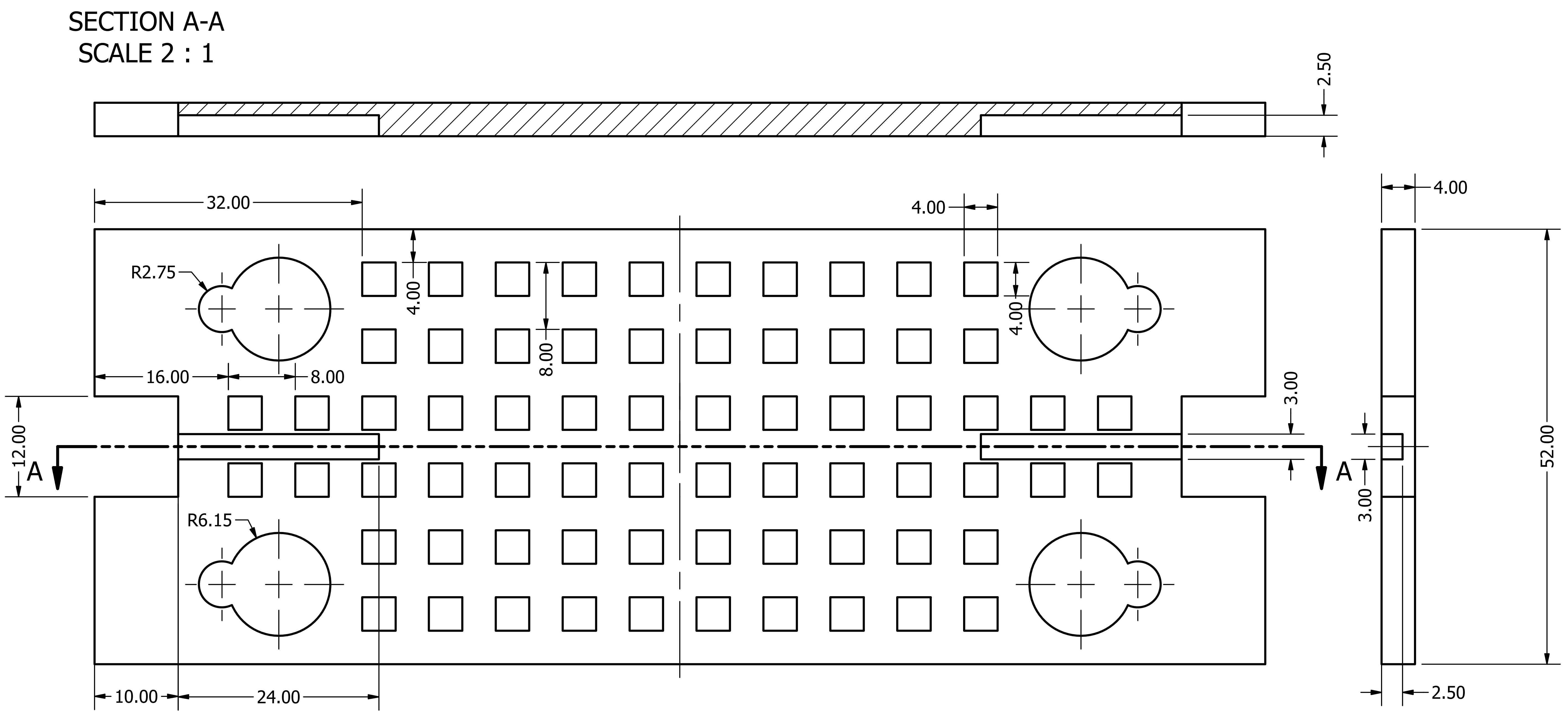
- Autodesk (2018) Inventor Professional 2018 – CAD Software https://www.autodesk.com/products/inventor/overview
- Shih, R.H. (2017) Learning Autodesk Inventor 2018 - Modeling, Assembly and Analysis, ISBN: 978-1-63057-131-3, SDC Publications https://www.sdcpublications.com/Textbooks/Learning-Autodesk-Inventor-2018/ISBN/978-1-63057-131-3/
- Jovanovic, Vukica (2014) Tiger face tutorial: Retrieved from https://www.youtube.com/watch?v=wCv3EwwYx3s

Acknowledgment: This work is supported by National Science Foundation grant 1749566

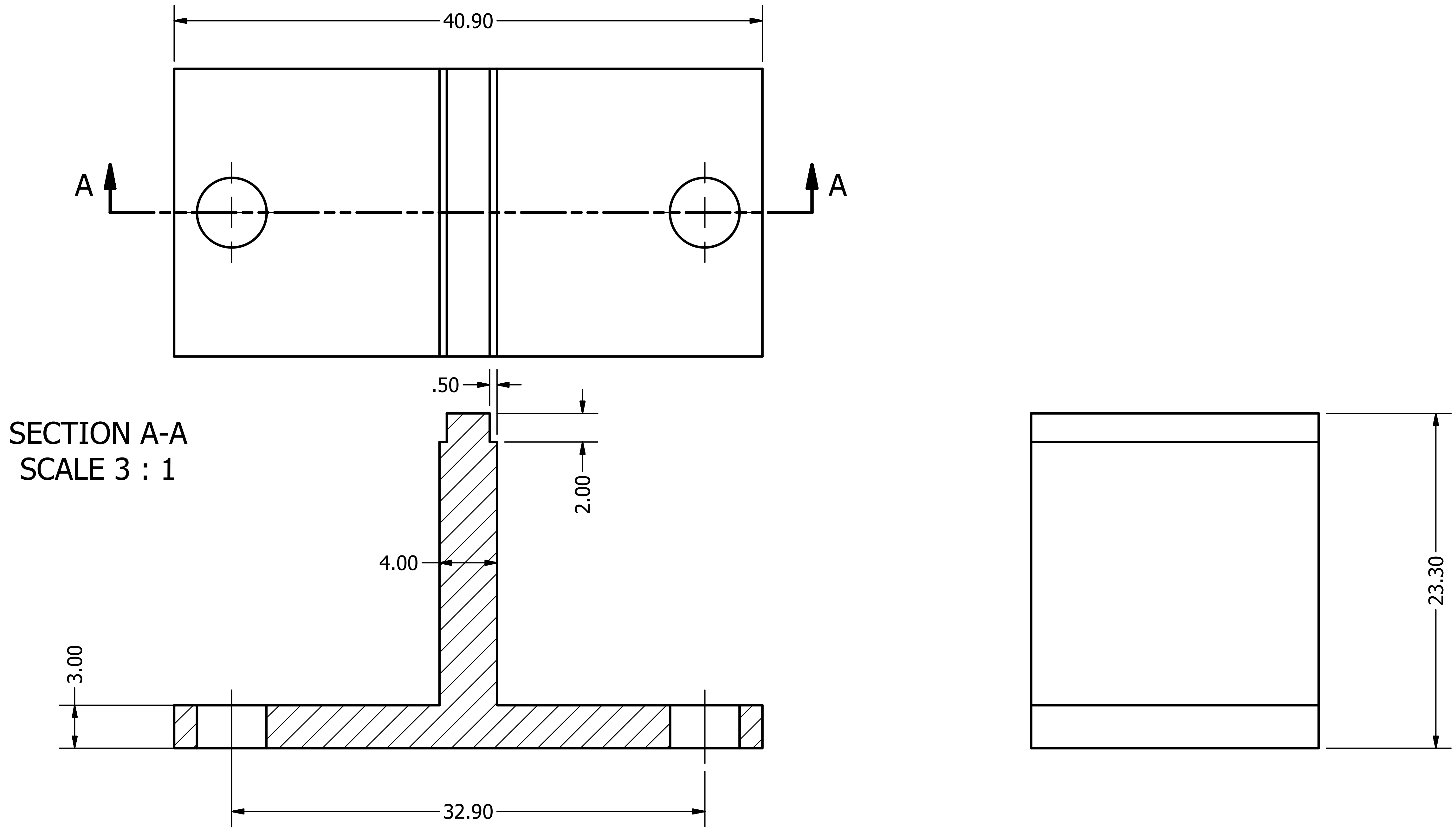


SECTION A-A
SCALE 3 : 1

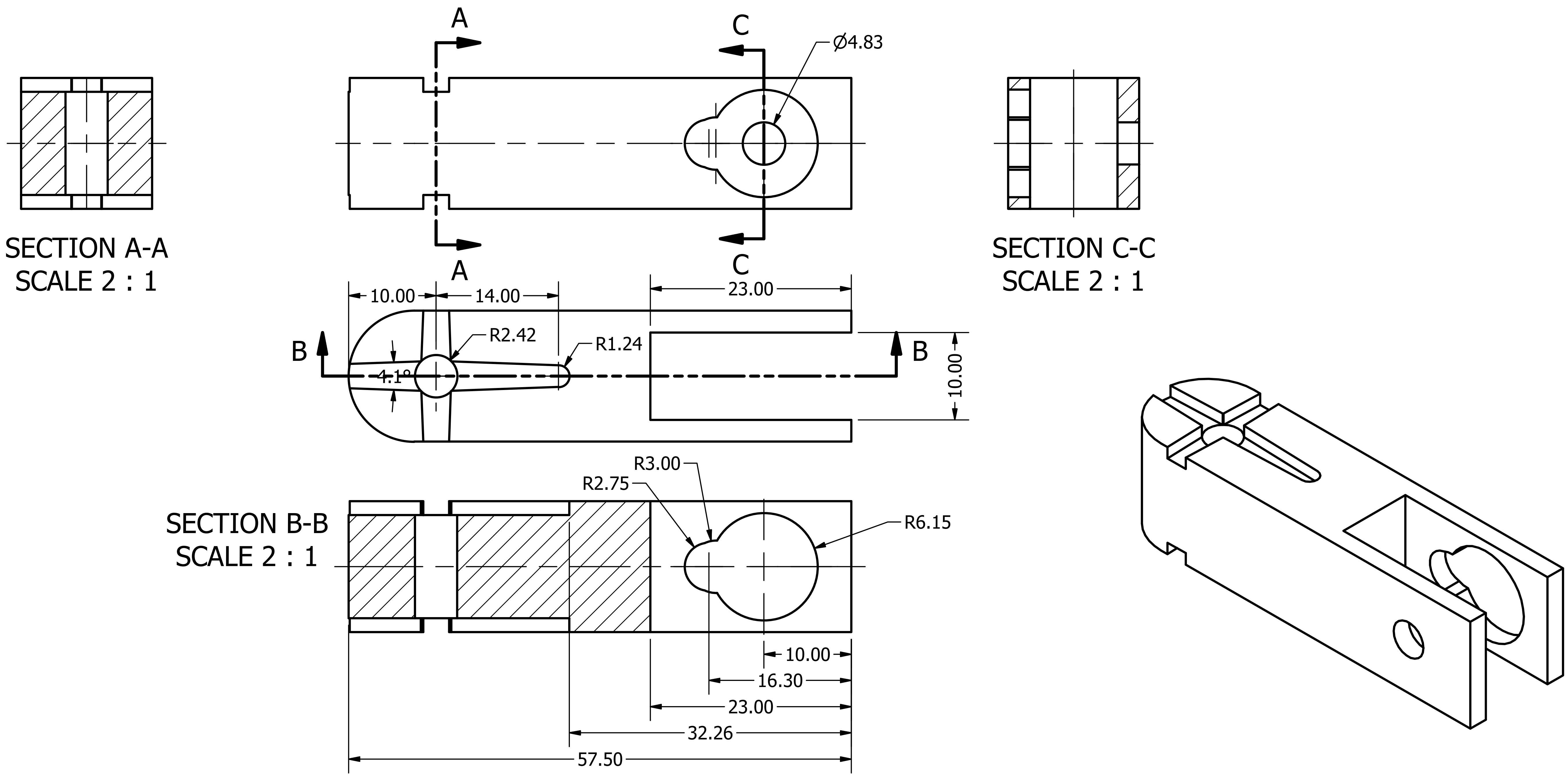
DRAWN V. Jovanovic	05/10/2018	Old Dominion University
CHECKED K. Kaipa	05/10/2018	TITLE
QA		Fore
MFG K. Arcaute	05/10/2018	
APPROVED M. Wang	05/10/2018	SIZE B · DWG NO 02 · REV
		SCALE 3 : 1 · SHEET 1 OF 1



DRAWN V. Jovanovic	05/10/2018	Old Dominion University			
CHECKED K. Kaipa	05/10/2018	TITLE			
QA		Body			
MFG K. Arcaute	05/10/2018				
APPROVED M. Wang	05/10/2018				
		SIZE B		DWG NO 01	REV
		SCALE 2 : 1		SHEET 1 OF 1	



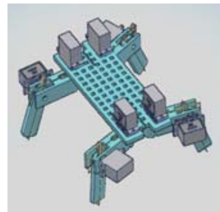
DRAWN V. Jovanovic	05/10/2018	Old Dominion University			
CHECKED K. Kaipa	05/10/2018	TITLE			
QA		Support			
MFG K. Arcaute	05/10/2018				
APPROVED M. Wang	05/10/2018	SIZE B		DWG NO 04	REV
		SCALE 3 : 1		SHEET 1 OF 1	

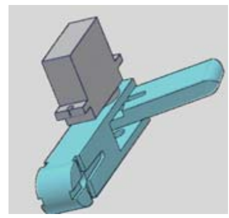


DRAWN V. Jovanovic	05/10/2018	Old Dominion University		
CHECKED K. Kaipa	05/10/2018	TITLE		
QA		Upper		
MFG K. Arcaute	05/10/2018			
APPROVED M. Wang	05/10/2018	SIZE B	DWG NO 05	REV
		SCALE 2 : 1		SHEET 1 OF 1

Module

Computer Aided Design (CAD)





Assembly Modeling and Motion Analysis

Acknowledgment: This work is supported by National Science Foundation grant 1749566

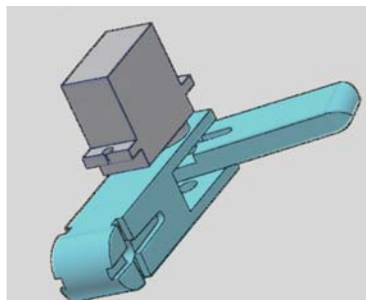
Assembly Modeling and Motion Analysis

- Understand the Assembly Modeling Methodology
- Understand and Utilize Assembly Constraints
- Understand the Autodesk Inventor DOF Display
- Utilize the Autodesk Inventor Drive Constraint Option
- Record Animation Movies

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Introduction

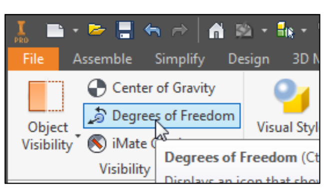
- To assemble parts into an assembly, we will need to consider the assembly relationships between parts.
- It is a good practice to assemble parts based on the way they would be assembled in the actual manufacturing process.

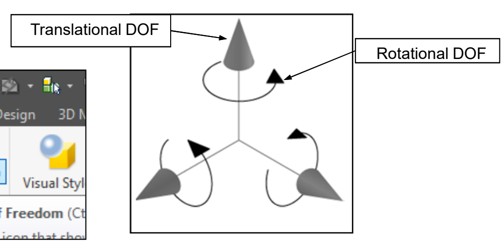


Acknowledgment: This work is supported by National Science Foundation grant 1749566

Degrees of Freedom and Constraints

- Each component in an assembly has six degrees of freedom (DOF), or ways in which rigid 3D bodies can move: movement along the X, Y, and Z axes (translational freedom), plus rotation around the X, Y, and Z axes (rotational freedom).
- Translational DOFs allow the part to move in the direction of the specified vector. Rotational DOFs allow the part to turn about the specified axis.

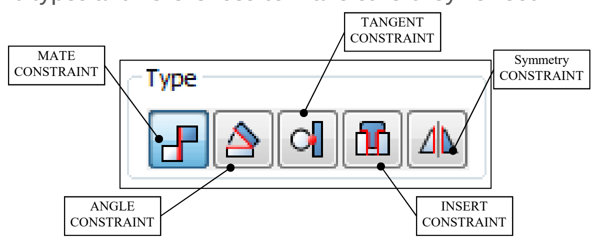




Acknowledgment: This work is supported by National Science Foundation grant 1749566

Assembly Constraints

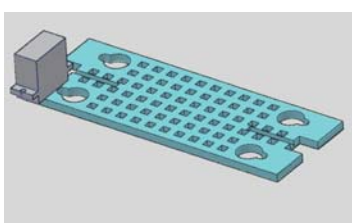
- To assemble components into an assembly, we need to establish the assembly relationships between components.
- It is a good practice to assemble components the way they would be assembled in the actual manufacturing process.
- Assembly constraints create a parent/child relationship that allows us to capture the design intent of the assembly.
- Because the component that we are placing actually becomes a child to the already assembled components, we must use caution when choosing constraint types and references to make sure they reflect the intent.

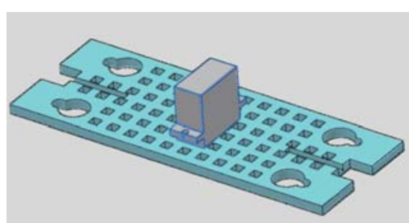


Acknowledgment: This work is supported by National Science Foundation grant 1749566

Constrained Move

- To see how well a component is constrained, we can perform a constrained move.
- A constrained move is done by dragging the component in the graphics window with the left-mouse-button.
- A constrained move will honor previously applied assembly constraints.
- That is, the selected component and parts constrained to the component move together in their constrained positions. A grounded component remains grounded during the move.

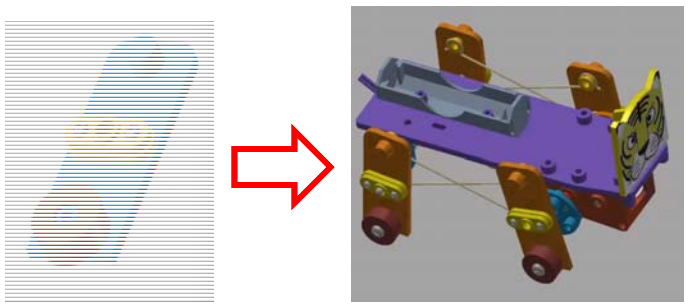




Acknowledgment: This work is supported by National Science Foundation grant 1749566

Subassembly

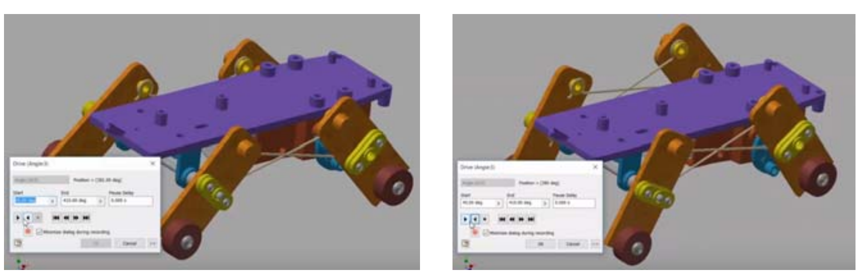
- We should also consider breaking down the assembly into smaller subassemblies, which helps the management of parts.
- In Autodesk Inventor, a subassembly is treated the same way as a single part during assembling.



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Motion Analysis

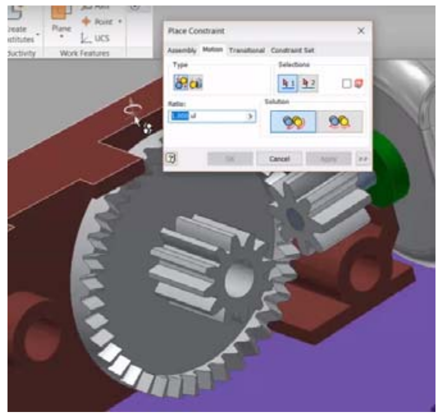
- Motion analysis can also be performed to visually confirm the proper assembly of the designs, also to check for any interference between mating parts and any other potential problems.



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Drive Constraint Tool

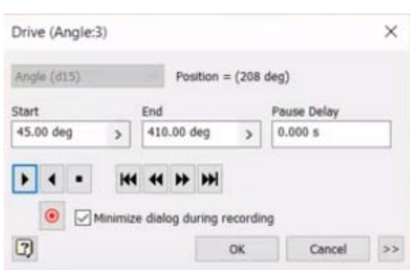
- In Autodesk Inventor, several options are available to perform motion analysis, for example, the Dynamic Simulation module, the Inventor Studio module and the Drive Constraint tool.

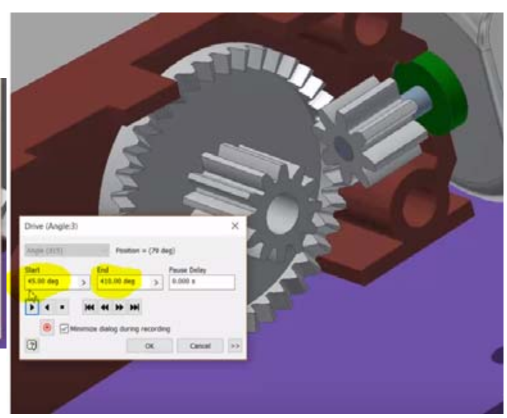


Acknowledgment: This work is supported by National Science Foundation grant 1749566

Drive Constraint Tool

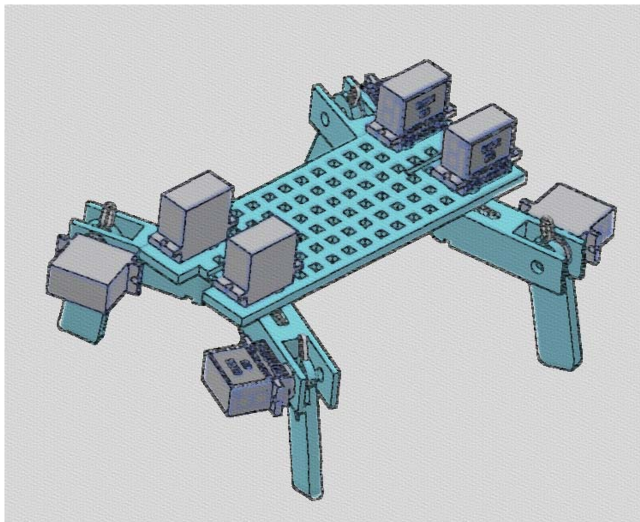
- The Drive Constraint tool provides a relatively simple motion analysis that can be done in a matter of seconds.





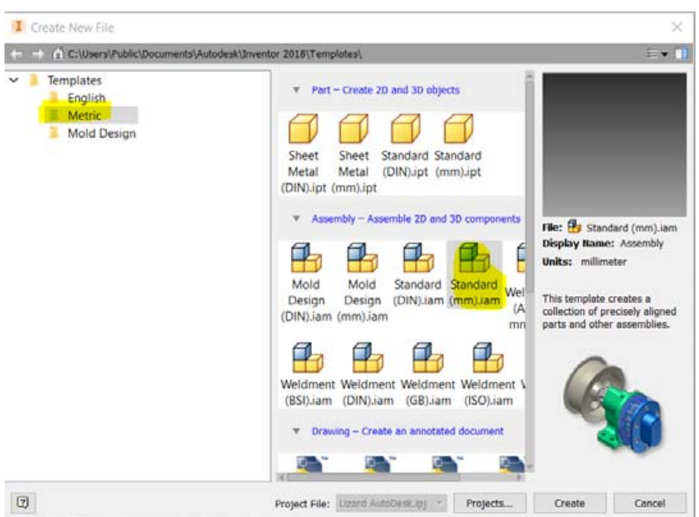
Acknowledgment: This work is supported by National Science Foundation grant 1749566

The Lizard Assembly



Acknowledgment: This work is supported by National Science Foundation grant 1749566

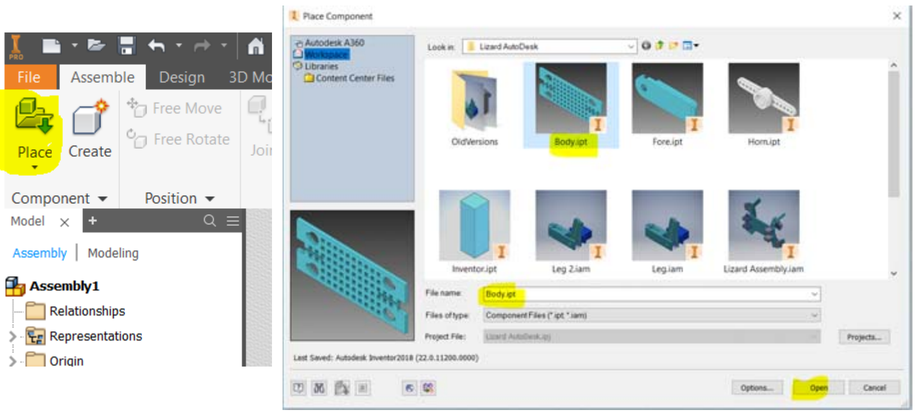
Create New Assembly (Metric)



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Placing the First Component

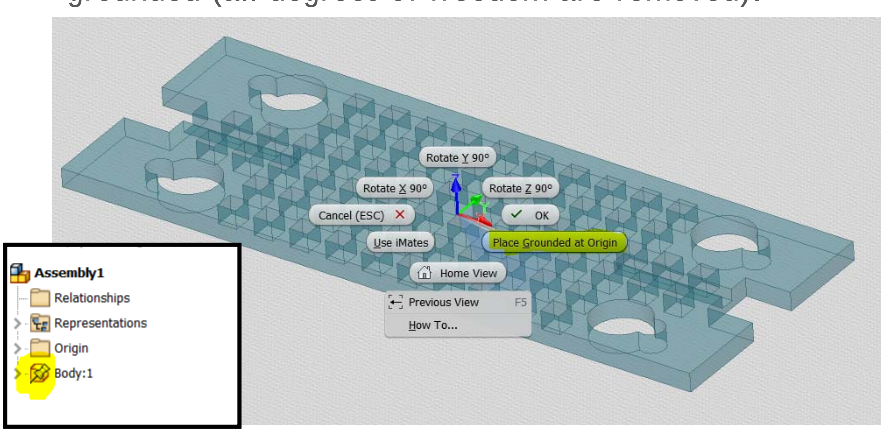
- The first component placed in an assembly should be a fundamental part or subassembly.



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Placing the first Component

- The origin of the first component is aligned to the origin of the assembly coordinates and the part is grounded (all degrees of freedom are removed).



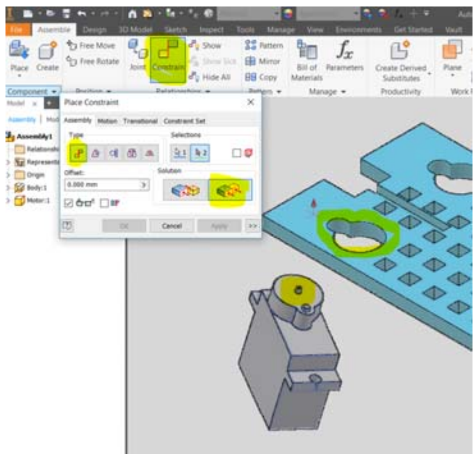
Acknowledgment: This work is supported by National Science Foundation grant 1749566

All Other Parts Will Be Constrained In Relation to the First Component

- The first component in an assembly file sets the orientation of all subsequent parts and subassemblies.

Constraint # 1

Two planes - Flush



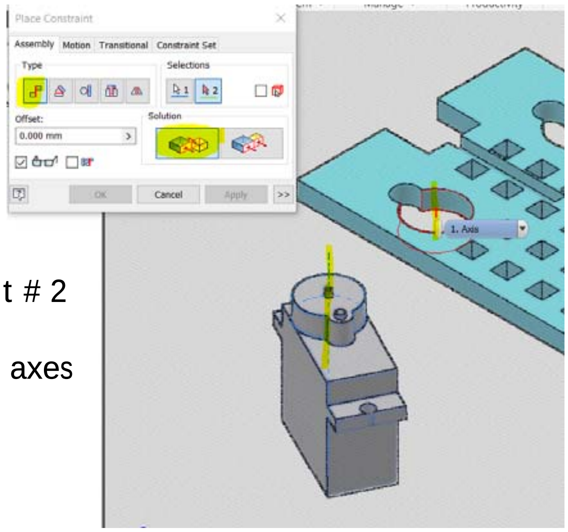
Acknowledgment: This work is supported by National Science Foundation grant 1749566

All Other Components are Constrained In Relation to The First Component

- The rest of the assembly is built on the first component, the base component. In most cases, this base component should be one that is not likely to be removed and preferably a non-moving part in the design.
- Note that there is no distinction in an assembly between components; the first component we place is usually considered as the base component because it is usually a fundamental component to which others are constrained.

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Placing the Second Component



Constraint # 2

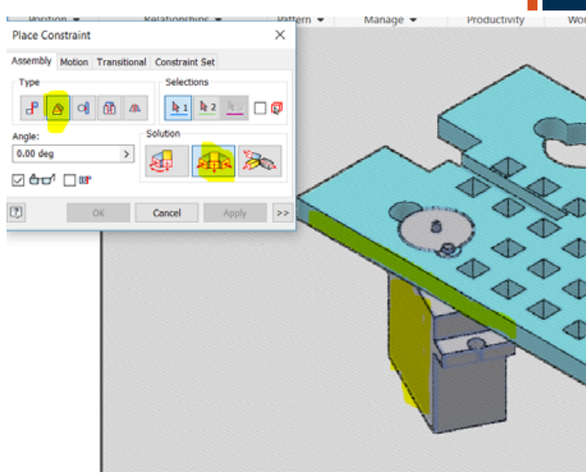
Align two axes

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Placing the Second Component

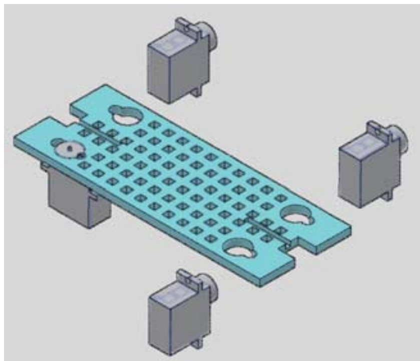
Constraint # 3

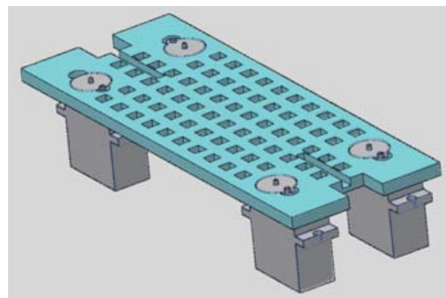
Angle = 0 degrees (parallel)



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Repeat for Other 3 Motors

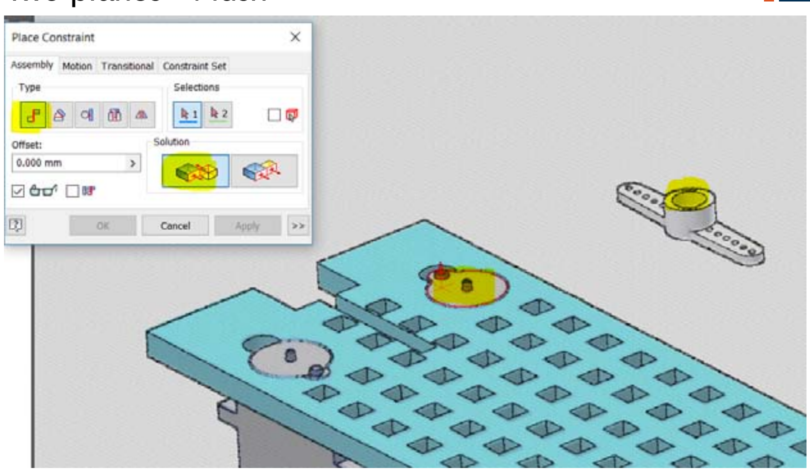




Acknowledgment: This work is supported by National Science Foundation grant 1749566

Placing the Third Component - Horn

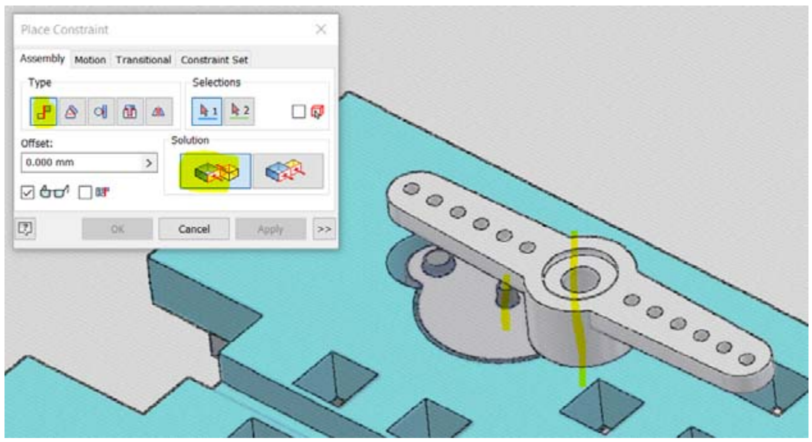
Constraint # 1 -
Two planes - Flush



Acknowledgment: This work is supported by National Science Foundation grant 1749566

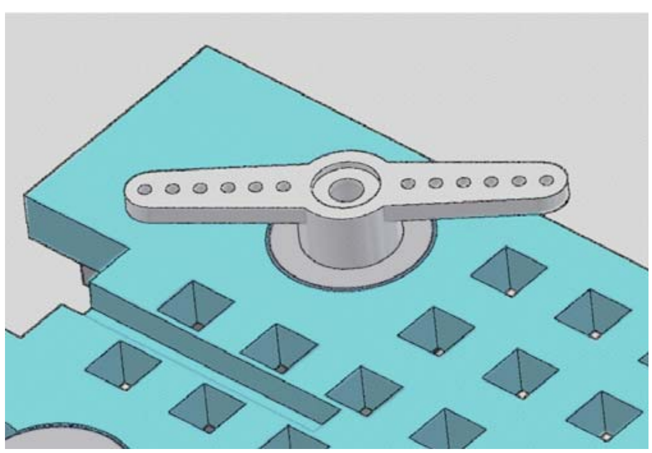
Placing the Third Component - Horn

Constraint # 2 -
Mate – Align Axis



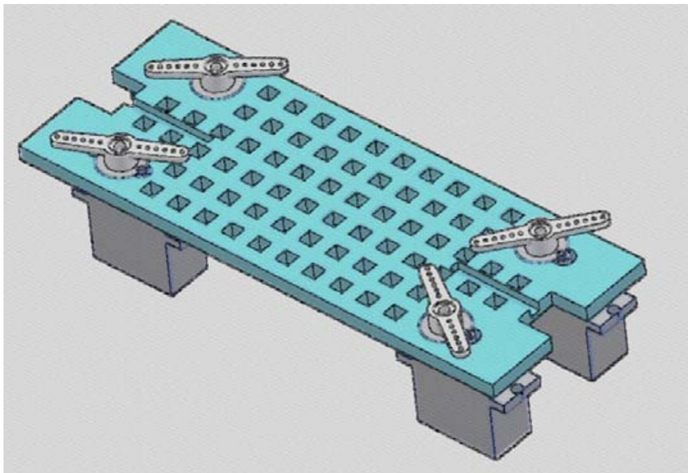
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Third One Should Be Free
Servo Motor Movement



Acknowledgment: This work is supported by National Science Foundation grant 1749566

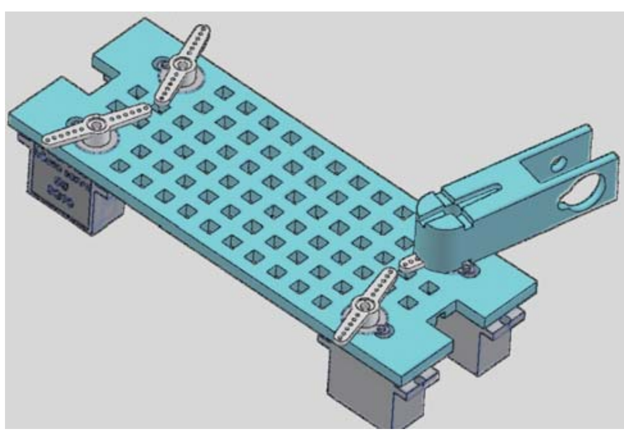
Do the Same for 3 More Horns



Acknowledgment: This work is supported by National Science Foundation grant 1749566

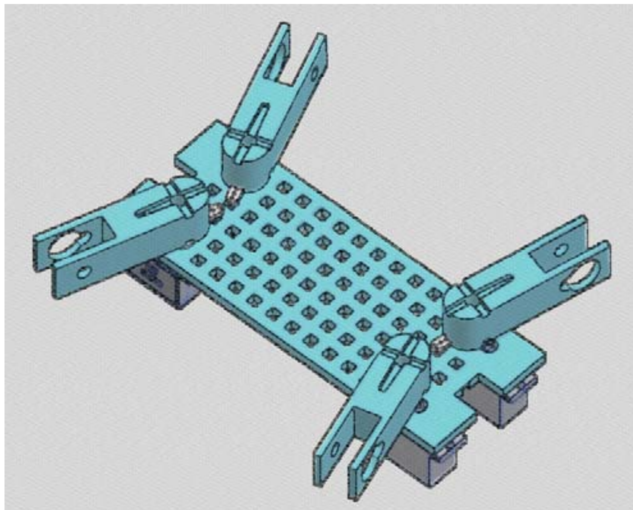
Add Upper Part

Constraint # 1 - Two planes - Flush	Constraint # 2 - Mate – Align Axis	Constraint # 3 - Unconstrained



Acknowledgment: This work is supported by National Science Foundation grant 1749566

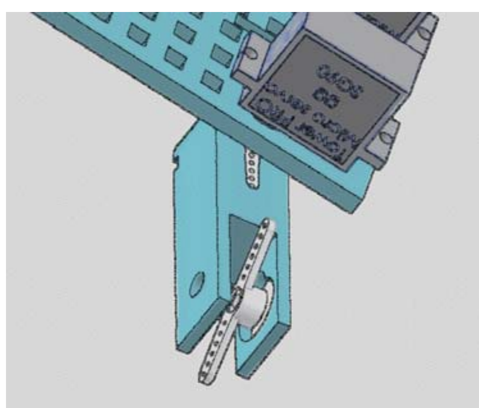
Add 3 More Upper Parts



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Position 4 Horns On Upper Parts

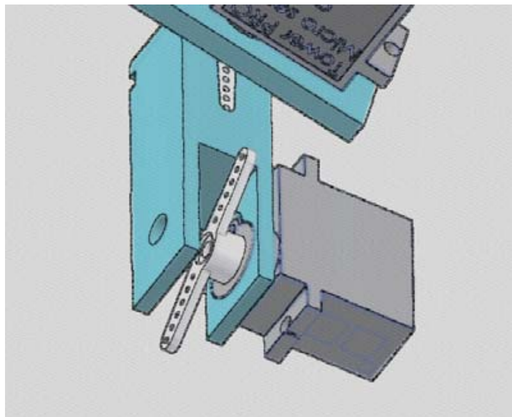
Constraint # 1 - Two planes - Flush	Constraint # 2 - Mate – Align Axis	Constraint # 3 - Unconstrained



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Position Motors at Upper Parts

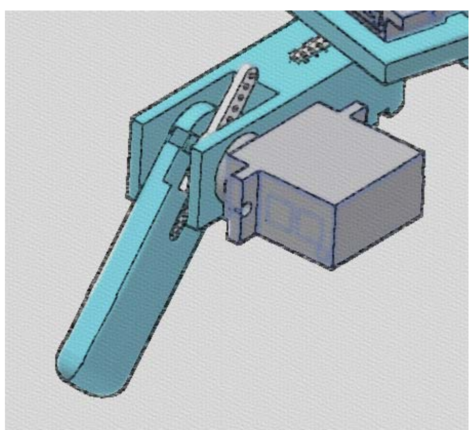
Constraint # 1 - Two planes - Flush	Constraint # 2 - Mate – Align Axis	Constraint # 3 - Angle = 0



Acknowledgment: This work is supported by National Science Foundation grant 1749566

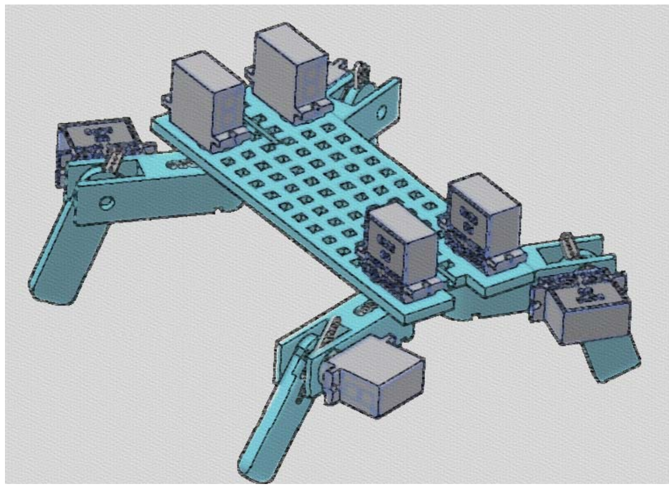
Add Part Named Fore

Constraint # 1 - Two planes - Flush	Constraint # 2 - Mate – Align Axis	Constraint # 3 - Flush Planes



Acknowledgment: This work is supported by National Science Foundation grant 1749566

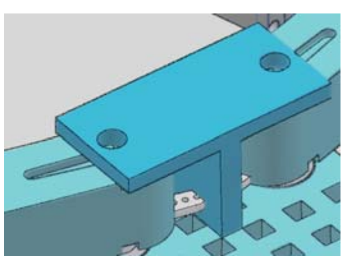
Add 3 Other Fore Parts

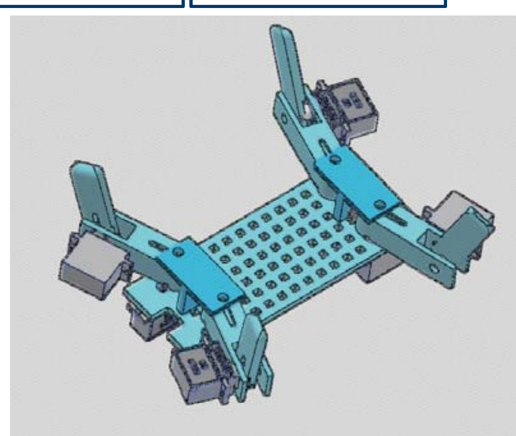


Acknowledgment: This work is supported by National Science Foundation grant 1749566

Add Support Part

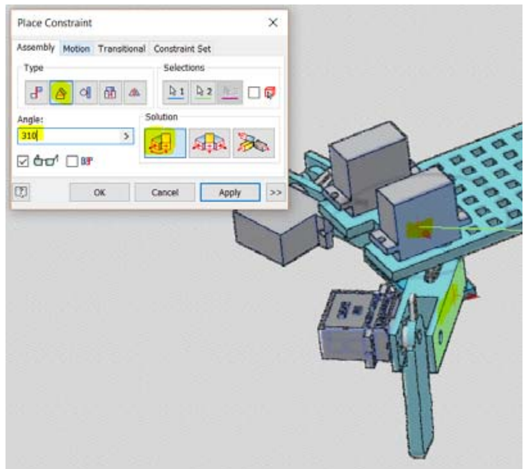
Constraint # 1 - Two planes - Flush	Constraint # 2 - Mate – Align Axis	Constraint # 3 - Mate – Align Axis





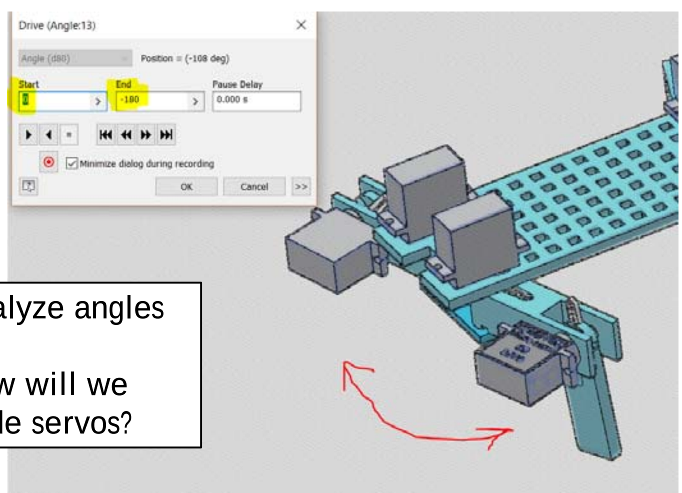
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Add Constraint Angle Between Motors and Upper Parts



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Motion Analysis

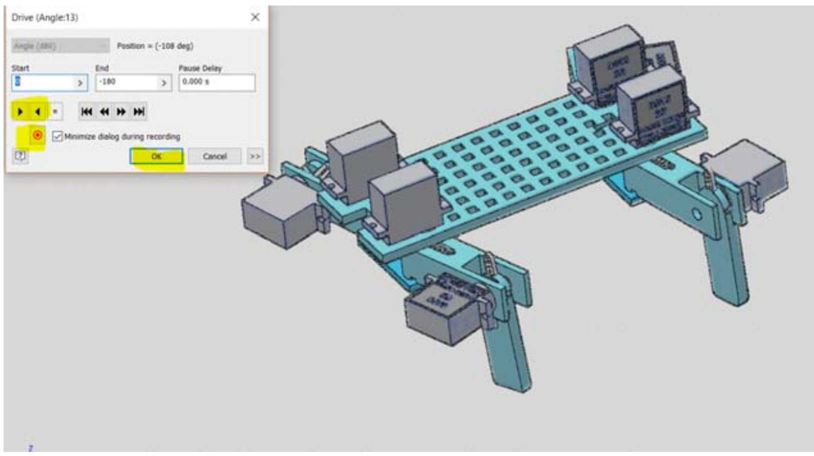


Analyze angles

How will we code servos?

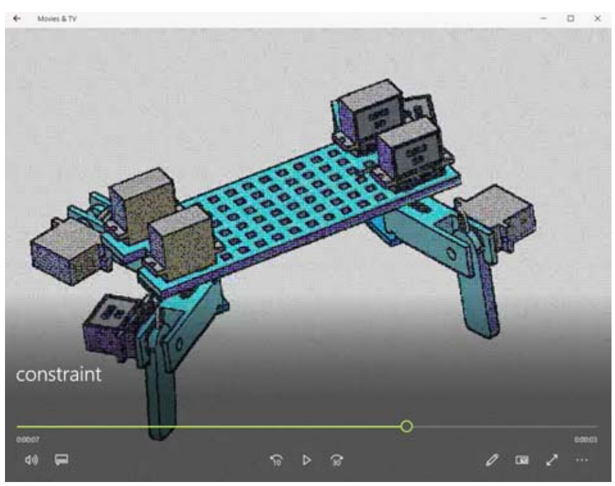
Acknowledgment: This work is supported by National Science Foundation grant 1749566

Record a Video



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Save the Video



Acknowledgment: This work is supported by National Science Foundation grant 1749566

Newport News Shipbuilding Launches the Digital Shipyard

35

Newport News Shipbuilding launches the digital shipyard

https://www.youtube.com/watch?v=93za-vO_ffs

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Digital twin: Making your asset smarter with the digital twin

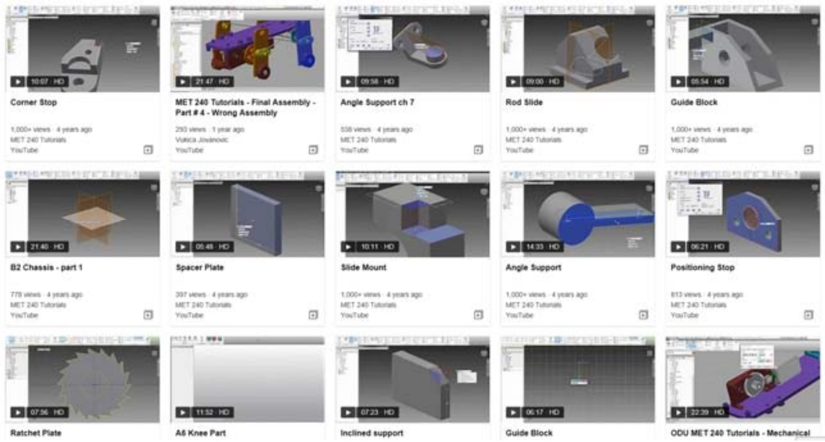
36

Digital twin: Making your asset smarter with the digital twin

https://www.youtube.com/watch?v=8ger1wrAonM

Acknowledgment: This work is supported by National Science Foundation grant 1749566

More Autodesk Inventor Tutorials Search for "MET 240 Tutorials"



Acknowledgment: This work is supported by National Science Foundation grant 1749566

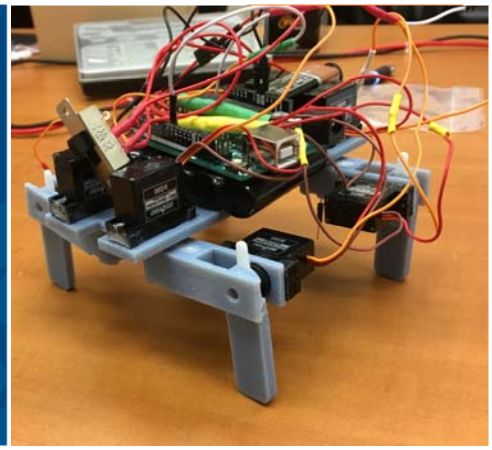
References

- Autodesk (2018) Inventor Professional 2018 – CAD Software https://www.autodesk.com/products/inventor/overview
- Shih, R.H. (2017) Learning Autodesk Inventor 2018 - Modeling, Assembly and Analysis, ISBN: 978-1-63057-131-3, SDC Publications https://www.sdcpublications.com/Textbooks/Learning-Autodesk-Inventor-2018/ISBN/978-1-63057-131-3/
- Jovanovic, Vukica (2014) MET 240 Final Project Videos, YouTube: https://www.youtube.com/watch?v=gs3y-N0fEFw&list=PLvXnbqkpWxCfUUImQFRND1LQRI5OFIAME

Acknowledgment: This work is supported by National Science Foundation grant 1749566

Module

Assembly of Bio-inspired Robot



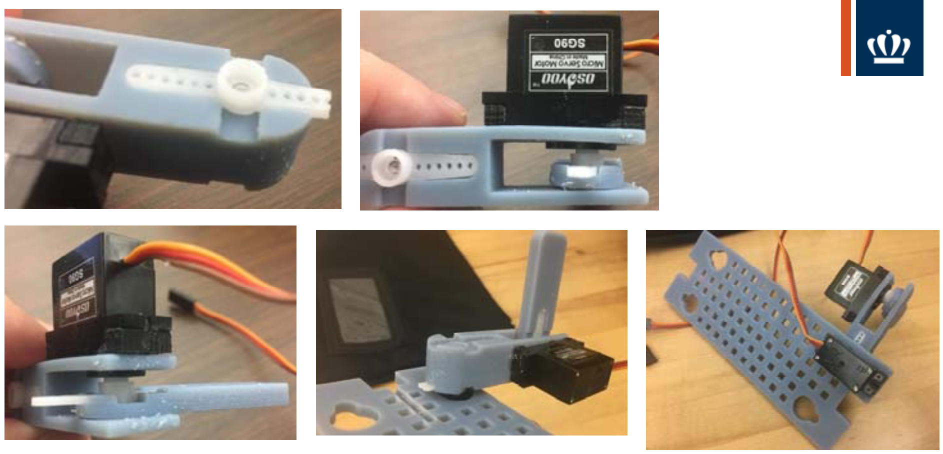
Assembly of Lizard-like four legged robot

Acknowledgment: This work is supported by National Science Foundation grant 1749566

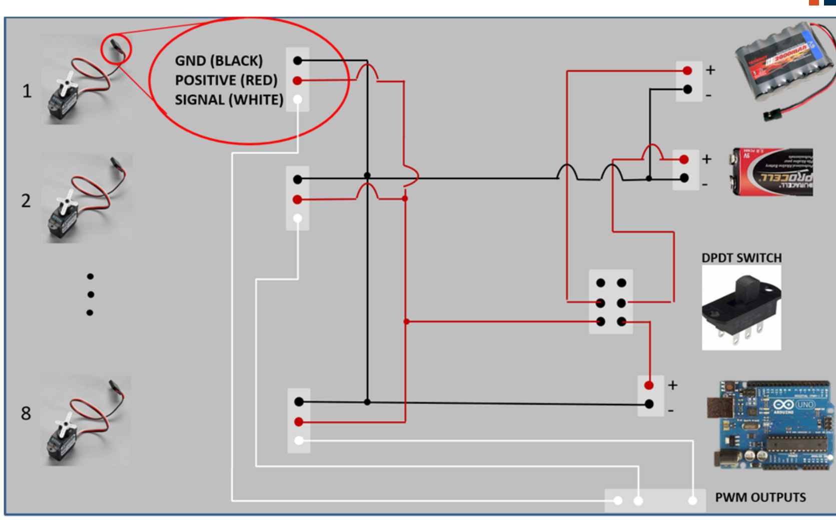
Assembly Steps

- Attach four servo motors to the main body
- Attach servo horns to all eight legs
- Attach servo motors to upper legs
- Run calibration code for each motor
- Attach legs to servo motors

Assembly Images



Electrical Circuit Schematic



Electrical Wiring