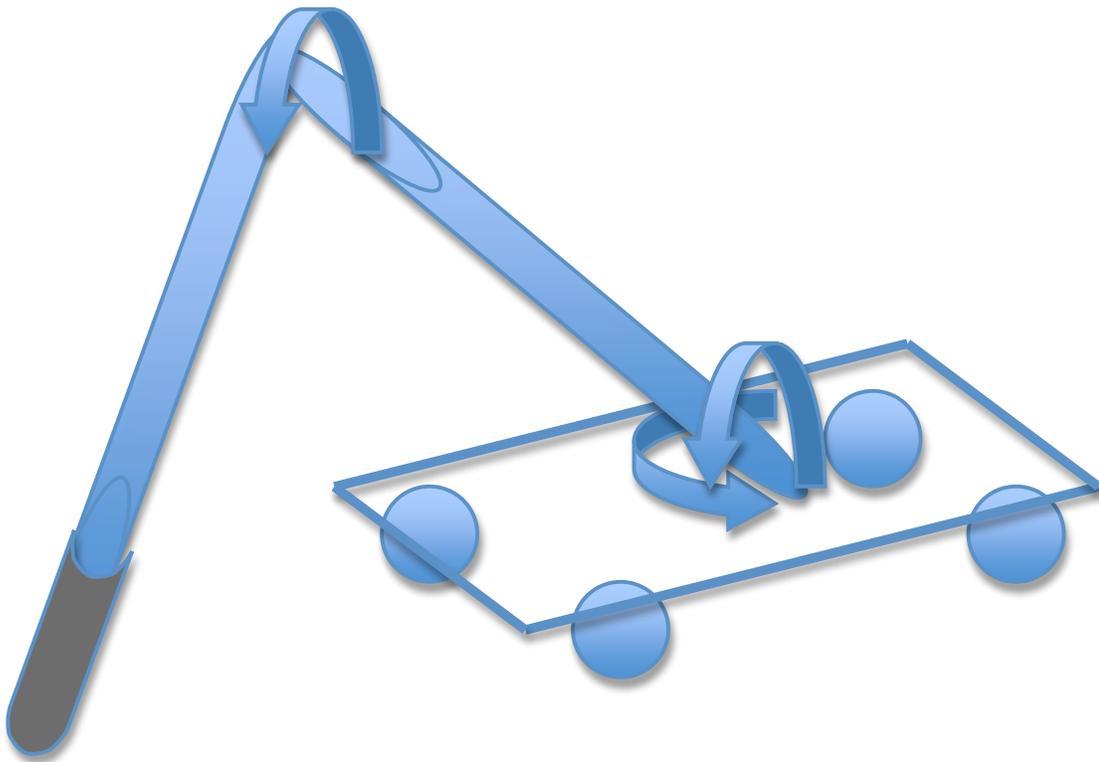


Design and Control of a Crawling Robot

By: Roby Velez



This is the documentation document for my project. The project is to design and control a crawling robot. This document will be constantly updated and uploaded to the wiki <http://wikis.swarthmore.edu/ECE90/index.php/Velez2009>.

Table of Contents

Update: 12/6/08.....	3
Setup Wireless XBee Connection	3
Talking between Computers.....	3
Listening to the pic.....	4
Talking to the pic	4
Troubleshooting.....	4
Plans for Servo Control Codes.....	4
Cool Mudskipper image	5
Update 12/8/08.....	5
Controlling the Pulse Width	5
Setting the Period	5
Setting Baud Rate	6
Operating PWM.....	6
Code for Servo Control	7
Problem with PWM.....	7
Things to do for next time:.....	7
Update 1/1/09	7
Debugging the servo control	7
Issue with pwm control.....	8
Use of timer1	8
Building the Chassis	8
Power issues	8
Future considerations	9
End affecter	9
Controlling the robot with XBee	9
Things to do for next time	10
Update- 1/12/09	10
Neural Network Control Architecture.....	10
Central Pattern Generators	10
Plan for controller development	11
Idea for a Tachometer	11

Update: 12/6/08

Setup Wireless XBee Connection

The wireless modules used are the XBee and XBee Pro 802.15.4 OEM RF Modules. The pro is going to be attached to the PCB since it is 5 volt tolerant and the pic runs off of 5 volts. Another module will be connected to a computer. Commands will be sent from the remote computer to the microcontroller via the wireless connection. This will give a command line, remote control of the robot.

Wireless modules configured using X-CTU. There are two modules, a XBee and a XBee Pro. The destination address of each module was set to the source address of the other module. The modules were connected to two computers, with X-CTU installed.

Talking between Computers

Steps:

1. Download X-CTU from maxstream site.
2. Connect both modules to different computers.
3. Test/query with X-CTU (Write down the modem number for each).
4. Go to configuration and set the destination of each module to the source of the other. (SH->DH; SL->DL)
5. Go to the terminal tab on both computer and type something. It should appear on the terminal window on the other computer.

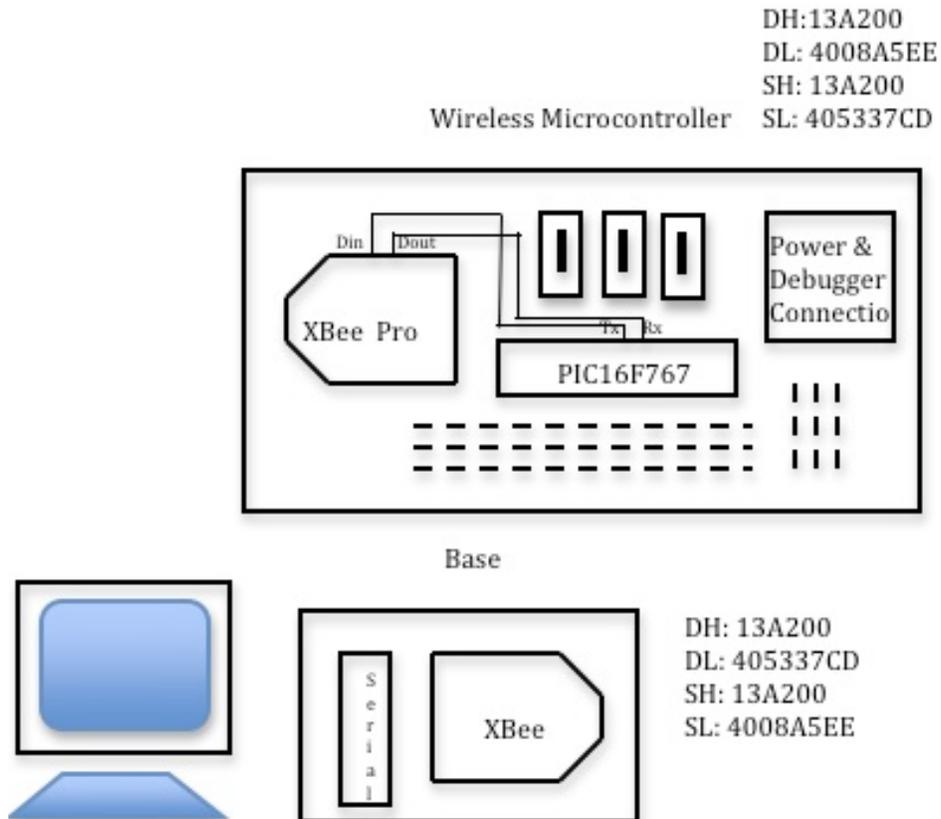


Figure 1: Diagram of the XBee and pic setup.

Listening to the pic

The PCB was designed for the Rx, and Tx of the pic to connect to the DI and DO pins of the XBee.

1. Configure the rs-232 to be a debugger. (#use rs232(debugger))
2. Write a simple program which allows the pic to print to othe debugger window.
3. Change the rs-232 setting to work with Rx and Tx pins. (#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8))
4. Disconnect XBee Pro from computer and connect Rx line of pic to DIN line of XBee.
5. Run code on pic.

The value originally printed to debugger will be sent through the Tx line into the controller XBee. The controller XBee will then transmit this to the base XBee, and the result will be displayed on the terminal.

Talking to the pic

1. Connect LED's to the pic and write code which will light the LED's in response to key presses.
2. Test with debugger if needed.
3. Change the rs-232 settings to work with Rx and Tx ahead if the debugger was used to the test the LED code.
4. Run the code on the pic.
5. Enter the commands in the X-CTU terminal connected to the base XBee. They should be sent to XBee. The base XBee will transmit them to the Controller XBee. It will then transmit the signal into the Rx pin of the pic. The pic will receive the commands and light up the LEDs.

Troubleshooting

1. Make sure the destination of each module is set correctly.
2. There was some odd with my board in that if I connected the Rx pin of the pic to the DIN pin of the XBee, while it was on the development board, the XBee would only transmit if the development board was connected to the computer via the serial cable. It turns out that as long as the shell of the serial cable was touching the development board it would transmit correctly. This is obviously wrong. Either there is something wrong with that particular board or they are designed incorrectly. What the XBee Pro was attached to my PCB board for real it worked, and transmitted just fine.
3. Used the receive and transmit LEDs above the power LED when debugging. They are very helpful.

Plans for Servo Control Codes

Pass in a value or time duration and have the servo go there.

- Need to pass in float, do float calculation, and convert answer back to int.
- Float calculation needs to be done with floats, ints, and longs.

Hold down u and it goes ccw. Hold down d and it goes cw. Also be able to set the speed.

- At first I can use a simple increment value. I've actually already done this.
- Use a float increment value and get the program to round up or down when evaluating the value or pulse width.

Rotate POT and the servo rotates too.

- Feed in some POT to the new board
- Scale the POT values to the value set to the PWM.
- There is going to have to be some type of float, int and long calculation and the answer will need to be rounded and made an int to be sent to the pwm function

Set a speed and have the servo go from origin to end of travel. And then reset to origin.
-Can I enter input that is more than one character long. But a certain speed needs to be scaled into an appropriate pwm value. It's going to be similar to the POT test. I want to be able to enter int values and float millisecond pulse widths

Cool Mudskipper image

<http://www.dkimages.com/discover/previews/995/55046589.JPG>

Update 12/8/08

Controlling the Pulse Width

Now it's time to try and control the servos.

Setting the Period

Servos require a pulse duration of 20ms to operate properly. 20ms translates into a frequency of 50Hz. In order to accomplish this the internal clock of the pic was set to 5KHz.

```
setup_oscillator(OSC_500KHZ|OSC_INTRC);
```

And timer2 which is the time base for the PWM was set to:

```
setup_timer_2(T2_DIV_BY_16,156,1);
```

The following equation sets the frequency.

$$\left(\frac{\text{OscillatorSpeed}}{4}\right)\left(\frac{1}{(\text{divisor})(\text{prescaler})}\right) = \left(\frac{500000}{4}\right)\left(\frac{1}{(16)(156)}\right) = 50 = \frac{1}{.02}$$

The help section of CCS C Compiler gives more a detailed description of the divisor and prescaler and what range of numbers are permitted. That combined with page 85 of the PIC16f767 datasheet helps provide an understanding of what the prescaler and divisor are doing.

Setting Baud Rate

5KHz seems to be quite slow for the computer and therefore a baud rate of 9600 couldn't be used. I had to set the baud rate on the pic to 2400.

```
#use rs232(baud=2400,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
```

I then had to set the baud rate on the two XBee modules. That entailed placing each module on the development board and issuing the follow commands. The commands need to be entered fairly quickly.

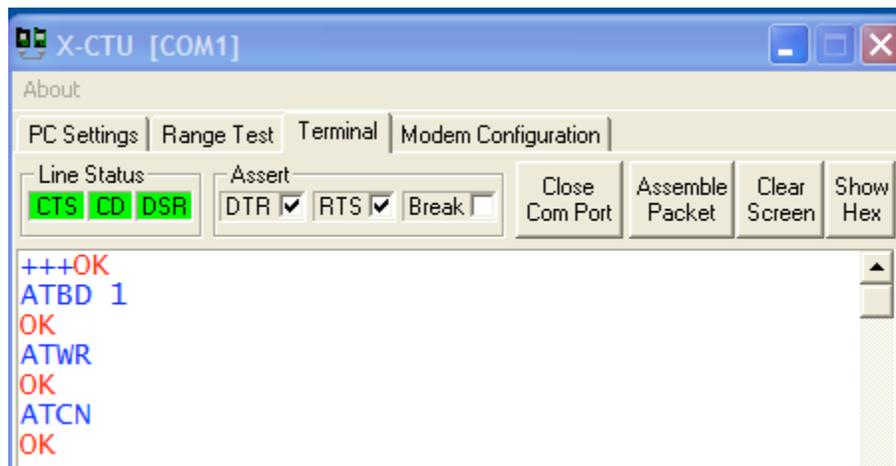


Figure 2: Scree shot of X-CTU terminal while setting baud rate.

```
+++ //Opens up communication  
ATBD 1 //Sets the baud rate to 2400  
ATWR //Write the configuration to the module  
ATCN //Closes the communication
```

All this is in the XBee module manual.

At the PC Settings if the baud rate is set to 2400 and a test/query is done it should come back successful. This needs to be done for the base pic and the one of the microcontroller.

Operating PWM

To operate the PWM first they must be declared.

```
setup_ccp1(CCP_PWM);
```

And set.

```
set_pwm1_duty(pwm_val);
```

$$pwm_val = pulseLength / (divisor * (1 / Fosc))$$

For the current settings a pulseLength of 1ms, or 5% duty produces:

$$pwm_val = (.001 * 500000) / 16 = 31.25$$

Code for Servo Control

Code was written which:

- Set the PWM value and then incremented or decremented by some value.
- Accepted a number in micro seconds and set the pulse width to this value.
- Accepted a speed, set the pulse width to zero, and then increment the pulse width to some maximum value by this speed. The bigger the speed the faster the increment.
- Read a potentiometer and mapped a range of pulse width to the potentiometer voltage.

IncrementServo.c
SetPosition.c
setSpeed.c
CommandviaPOT.c

Problem with PWM

When the servos are connected to the pic and these programs are executed they don't perform as expected. The width of the pulse changes as it should, but the servo doesn't cooperate.

Things to do for next time:

- Debug servos and get them to work more reliably.
- Test the low power pic and see if it works with the servos.
- Begin looking at designing the chassis.

Update 1/1/09

Debugging the servo control

Issue with pwm control.

The servos were found to operate from 500us to about 2500us. If these values are substituted into the equation for pwm_value shown above:

$$pwm_val = (.0005 * 500000) / 16 = 15.625$$

$$pwm_val = (.0025 * 500000) / 16 = 78.125$$

The difference between the two pwm values is 62.5. Because the c code uses integers this means the max resolution is 62 increments. The servos can rotate 180 degrees. This leads to about 3 degrees of movement per increment.

When a potentiometer was connected to the servo, the movement was very jittery.

Use of timer1

Timer1 is a 16 bit counter. When it overflows it can initialize an interrupt. Timer 1 was made to overflow/interrupt after 20ms.

The pwm of the pic have three modes. The first is to produce a pulse of a certain width. This has already been tried. Another mode of the pwm is to launch an interrupt once timer 1 has reached a certain value, pwm_cmp.

What can be done is to set the output of pin_c1 to high, and start timer 1. When the value of timer 1 reaches pwm_cmp an interrupt for pwm1 will be launched. Once this happens the interrupt for pwm1 will set pin_c1 to low. Finally when timer 1 overflows it will set pin_c1 to high. This in affect creates a pulse of a certain width.

I don't have access to this equation right now, but will put it in the next update.

The viable range of control for the servo is 500 to 2500 microseconds. Using timer 1 this leads to a resolution of 4000 steps for the servos. With this system the servos are much smoother even are very low speeds.

Building the Chassis

Some servos and aluminum arms were scavenged from the E5 projects. The arm came together in very little time. Curious to see how far it could be taken castors and a piece of plywood was obtained from the shop.

Power issues

With the chassis built a preliminary test was conducted to see if I could control all three servos. A program was made to read in three potentiometer readings and map them onto the three pwm. This is using the timer 1 system. What was first observed was that when ever two or more servos were connected to the board the robot will lose control and just flail around violently.

Power comes into the board through a screw terminal. This power source supplies the microcontroller and the servos. For my initial setup I was a bread board to power the board. When more than one servo was attached they drew more current than the bread board could produced. This browned out the pic and caused it to reset. This caused havoc with the servos.

One solution was to power the servos off a separate bread board. This worked great, but the mass of cables was very messy. A second solution, suggested by Prof. Cheever was to use a power source that could produce more current.

A 2 amp power source was connected to the robot and the servos and microcontroller seemed to work fine.

Future considerations

Right now the beefier power supply seems to work, but this is not how the robot will operate. The robot will be mobile so it will have to utilize a battery. An issue is that if the battery's power drops below a certain point the pic will start to brown out and act erratically again. A possible solution is to have the microcontroller monitor the power source. If the power source reaches a certain level where operation would become erratic than the robot shuts down.

Another option is to have two power supplies. One for the servos and one for the microcontroller. We'll see how the single power supply, and battery monitoring system works out before this option is tried.

End affecter

Once the power issue was resolved enough to allow for a proper test, rubber bands were wrapped around the end of the arm. Using two potentiometers I was able to control the robot and have it successfully crawl. The robot will crawl.

Controlling the robot with XBee

Ultimately the robot will be controlled by sending commands via the wireless serial connection. I wanted to play wit the robot and see the extent of its crawling abilities, but attaching the potentiometers to the chassis as it is crawling around wasn't going to work.

First and foremost the connections weren't very secure. The wires would come out and the servos would flail. Therefore I wanted to attach the potentiometers to the base XBee and see if I could send the values via the wireless communication.

This is different from typing a command into Matlab or X-CTU. My initial efforts failed, even though I've found people online who have done the exact thing I was trying to do. Since I was doing this the day before I left for break I had to leave it at that.

My plan of attack now is first concentrate on reading in analog and digital inputs to the base XBee controller. If I can do this than I think sending the data will be simple. As of right now I don't even know if the base XBee module is reading the potentiometer or buttons attached to it.

Things to do for next time

- Research on neural networks
- Formalize a plan for when I get back to Swat
- Start thinking about the other sensors such as the force and tachometer
- insert the missing formals and pictures above

Update- 1/12/09

Neural Network Control Architecture

I want to somehow use a neural network to control the crawling, because then maybe I could implement some type of learning with it.

Central Pattern Generators

I think I know how I will control the servo arms. There is an area of research called Central Pattern Generators CPG. They work on the basis that locomotion can be seen as actions, which are cyclical in position. The servos start in an initial position, move through their range, and then return to that initial position. A neural net can easily be made into a CPG. In order to control the CPG you specify the frequency, set points, phase, and amplitude of the controller. If a CPG is attached to each servo than by altering this patterns many different behaviors can be achieved.

Researchers have already used CPG's to make a fish robot with not only swims, but crawls. Granted the crawling isn't as complex as my robot, but the project shows that CPG's can be used to coordinate multiple motors. They also used a gradient descent search algorithm to find parameters, which will make the fish crawl/swim fast and efficient.

Papers I've found:

Crespi, Alessandro, "Controlling swimming and crawling in a fish robot using a central pattern generator"

Crespi, Alessandro "Online Optimization of Swimming and Crawling in an Amphibious Snake Robot"

Plan for controller development

Controller 1: This controller will utilize a finite state machine, and the forward and inverse kinematics to move the robot.

Controller 2: The forward and inverse kinematics will be replaced by CPG's.

Controller 2a: I want to try to use some type of learning mechanisms to determine the parameters of the CPG.

Controller 3: I want to try to replace the finite state machine with some type of neural net which will grow.

Idea for a Tachometer

I think I can make a simple tachometer can be made with a large marble, marker, LED, and photoreceptor. The photo below shows a concept drawing and a prototype I made from an old roll on deodorant bottle.

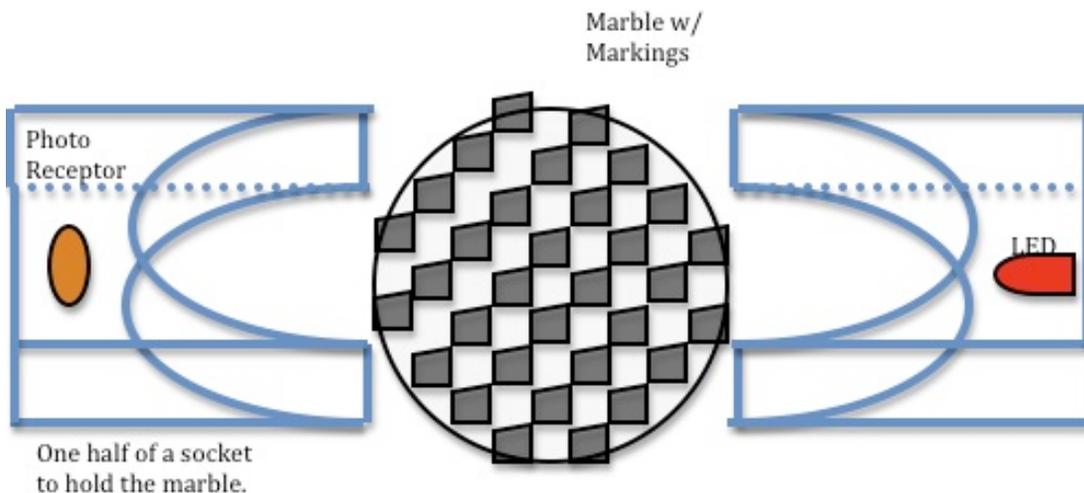


Figure 3: Concept art of homemade tachometer.

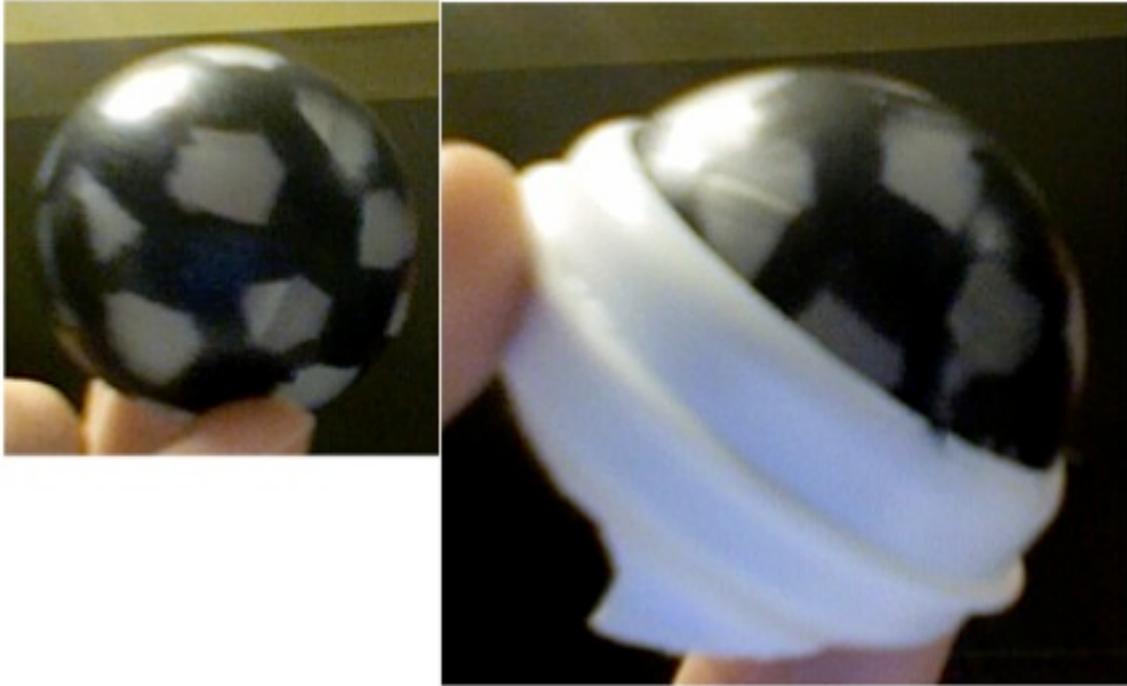


Figure 4: Prototype of Homemade Tachometer

The marble will be marked with paint, most likely in some checkerboard pattern. A socket will be made in which the marble can sit. In one side of socket there will be an LED. On the opposite side of the LED there will be a photoreceptor. As the marble spins in the socket the marked bands will get in the way of the light from the LED and cause the signal received by the photoreceptor to change.

The socket can then be made to replace one of the casters of the robot. As the robot rolls along the signal measured by the photoreceptor will change. This will indicate that the robot is moving. This will serve as a simple tachometer. What may also happen is if the marks on the marble are evenly spaced there made be a correlation to how many times the signal from the photoreceptor changes per minute and how fast the robot is moving.

Things for next time (back at Swat):

- Send analog and digital signals from XBee to Pic
- See if tach prototype works with LED and photo sensor
- Make a attach a suitable end effector
- Control robot remotely