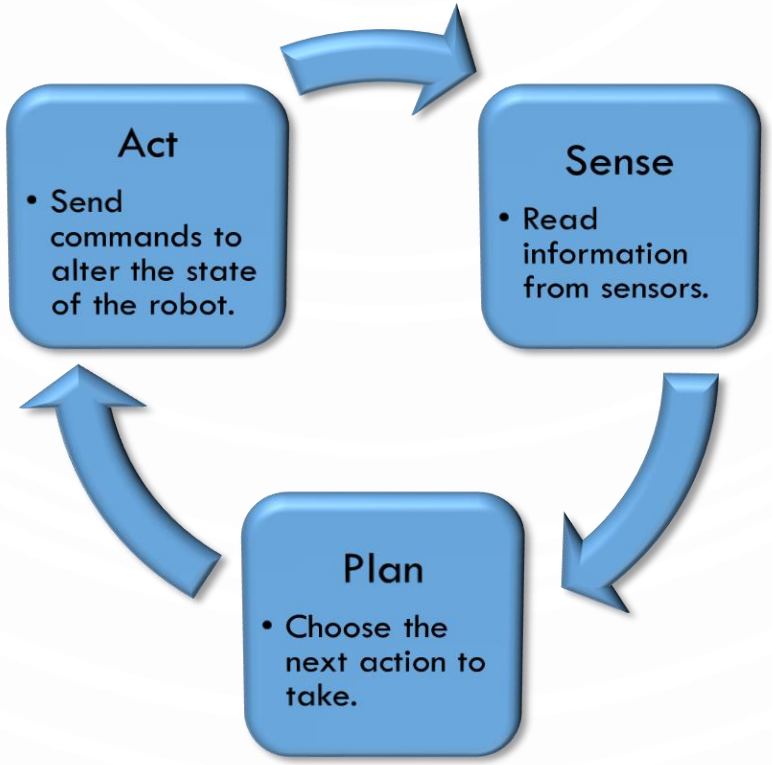
ROBOTICS
SENSE-PLAN-ACT LOOP

MOTIVATION

- In project 1 our robot couldn't return to exactly the same location as it started.
  - Why?
- We can solve this though continual feedback! We need to make decisions more often to adjust our actions.
  - Supports algorithmic thoughts like: drive **until** we see a wall, or turn **until** we face the right direction
- Essentially, we need a loop!

SENSE-PLAN-ACT LOOP

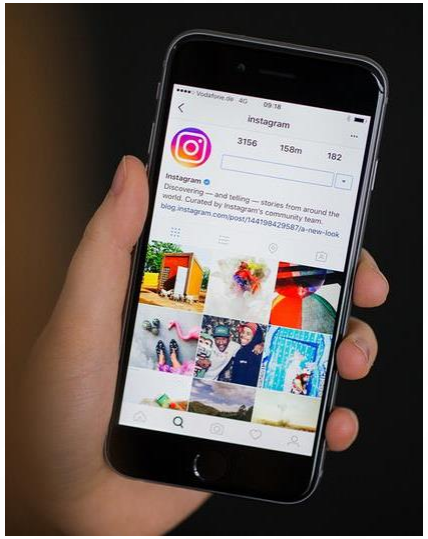


SENSE-PLAN-ACT LOOP

```
1. while robot_is_running():
2.     sense() # Read all sensor information
3.     plan()  # Make a plan and decide action to take
4.     act()   # Send commands to the robot
```

EVENT DRIVEN PROGRAMMING

- This loop is related to an approach for programming called **event driven programming**, which is extremely common in applications



- More generally for event driven programming:

```
while applicationIsRunning():
    processInputs()
    doSomethingAutomagically()
    provideFeedbackToUser()
```

- Tricky part is to alter your thinking to rely on this single loop to make things happen over time.

LETS THINK DEEPER ABOUT SOME ROBOT FUNCTIONS

- robot.forward()
  - Sets motors on
  - Continues application program immediately
  - Requires us to use time.sleep() to create motions

vs

- robot.drive_cm(x)
  - Set motors on for a set distance
  - Waits to continue application program until motion is complete
  - Can specify fully: robot.drive_cm(x, True)

True requires motion to finish, False continues program immediately.

FRAME LIMITING

- Many robots need some fixed form of "waiting" to pass the time before the program ends. We can include this in our loop:

```
1.  while robot_is_running():
2.      sense() # Read all sensor information
3.      plan()  # Make a plan and decide action to take
4.      act()   # Send commands to the robot
5.      wait()  # Wait for an amount of time, or to be more
                # sophisticated, wait for a remaining amount
                # of time based on fixed rate.
```

Work with a partner to alter the loop for the added sophistication.

EXERCISE

- Write a method that mimics a bumper but with a more complex aspect
  - Whenever an object is too close the robot stops and turns on a light
  - Whenever an object is too far the robot should move forward and turn off its light
  - The robot should continuously scan three different angles $45°, 90°, 135°$ for seeing if an object is too close. (1 reading per action taken by the robot)
  - After 30 seconds the program should terminate
  - The robot should make an action every 0.33 seconds.