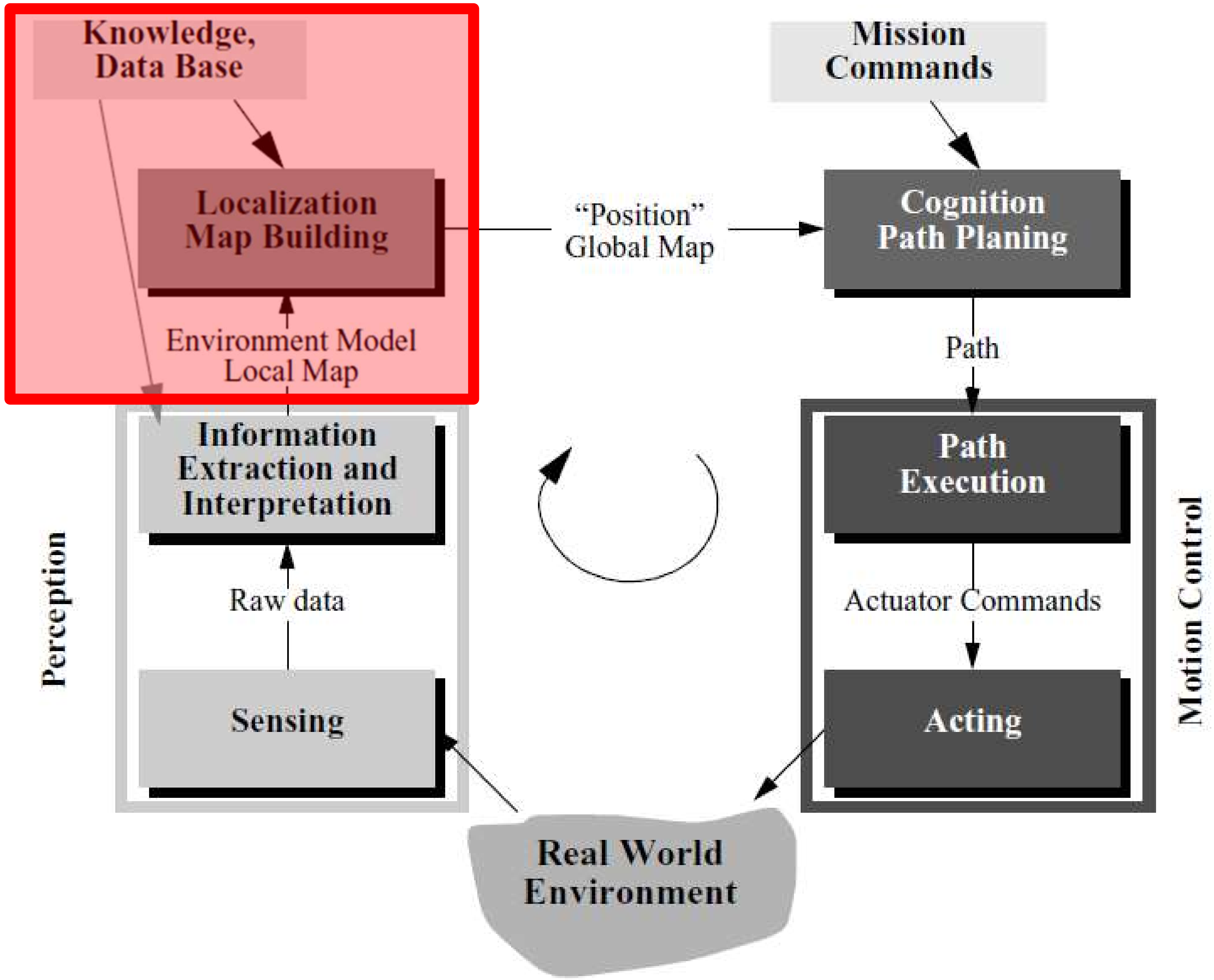
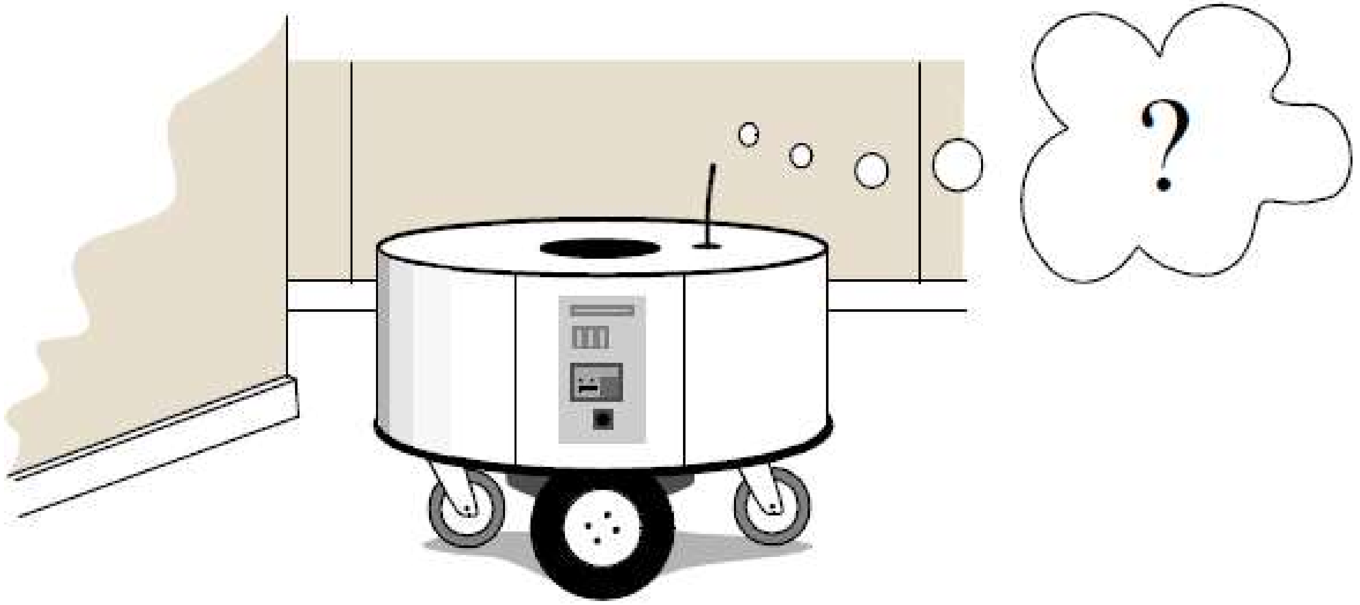
ENPM 809T

UMCP, Mitchell


Knowledge, Data Base
Localization Map Building
"Position" Global Map
Mission Commands
Cognition Path Planing
Path
Environment Model Local Map
Information Extraction and Interpretation
Raw data
Sensing
Perception
Path Execution
Actuator Commands
Acting
Motion Control
Real World Environment

Localization

- Robot must determine its position in its environment



Localization: Communications

- When event occurs:

1. Record image
2. Email image

Localization: Communications

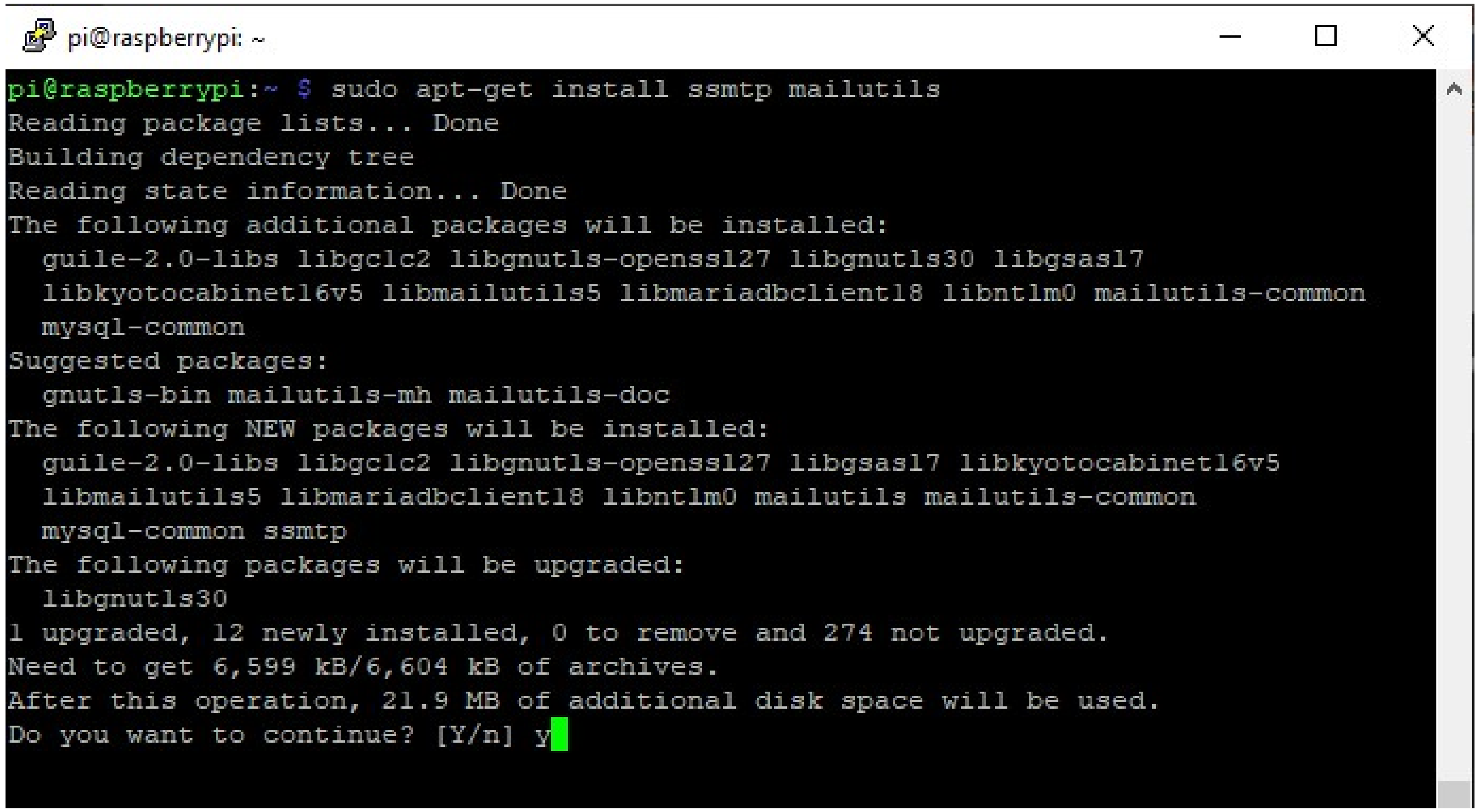
- Create a new email:

  ENPM809TS19@gmail.com

  Password: ************

Localization: Communications

- Confirm **ssmtp** & **mailutils** packages are installed



pi@raspberrypi: ~

```
pi@raspberrypi:~ $ sudo apt-get install ssmtp mailutils
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  guile-2.0-libs libgclc2 libgnutls-openssl27 libgnutls30 libgsasl7
  libkyotocabinet16v5 libmailutils5 libmariadbclient18 libntlm0 mailutils-common
  mysql-common
Suggested packages:
  gnutls-bin mailutils-mh mailutils-doc
The following NEW packages will be installed:
  guile-2.0-libs libgclc2 libgnutls-openssl27 libgsasl7 libkyotocabinet16v5
  libmailutils5 libmariadbclient18 libntlm0 mailutils mailutils-common
  mysql-common ssmtp
The following packages will be upgraded:
  libgnutls30
1 upgraded, 12 newly installed, 0 to remove and 274 not upgraded.
Need to get 6,599 kB/6,604 kB of archives.
After this operation, 21.9 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

If not installing properly:
run sudo apt update
then sudo apt full-upgrade

May also need to use:

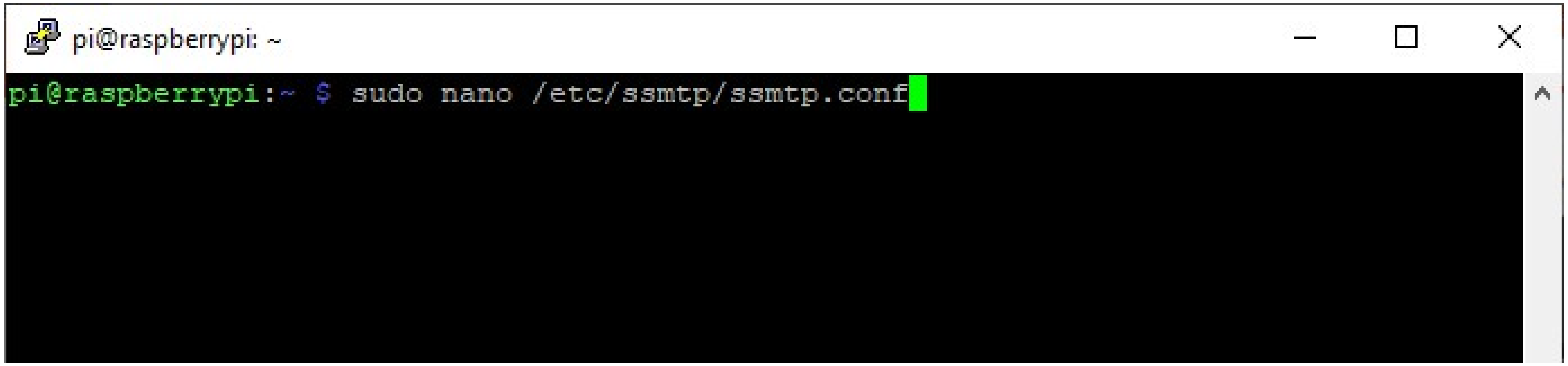
sudo apt update --allow-releaseinfo-change

https://www.youtube.com/watch?time_continue=13&v=0kpGcMjpDcw

sudo apt update --allow-releaseinfo-change

Localization: Communications

- SSMTP (est. 1982): program which delivers email from a local computer to a configured mailhost (i.e. mailhub)
- Edit the ssmtp config file

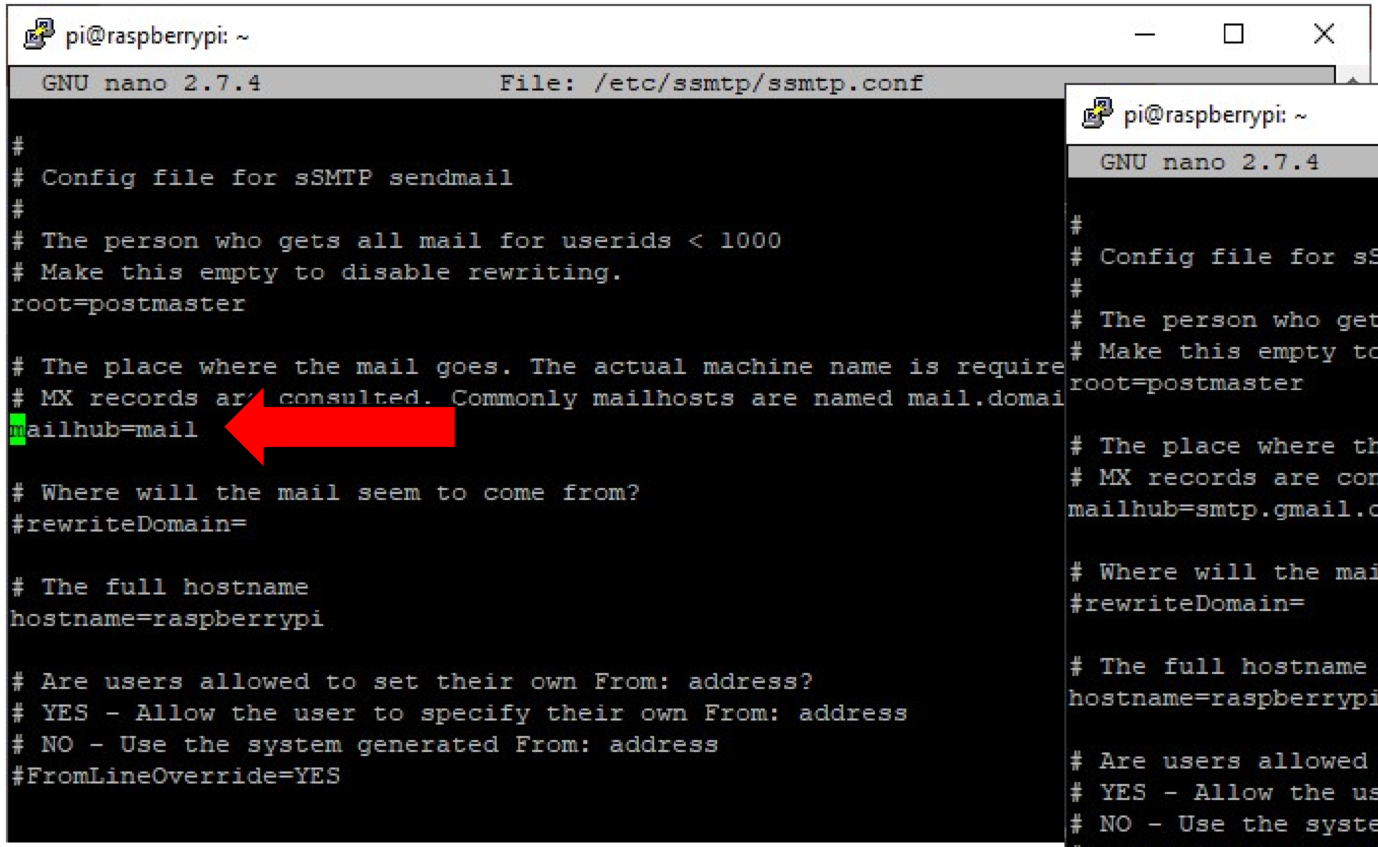


```
pi@raspberrypi:~ $ sudo nano /etc/ssmtp/ssmtp.conf
```

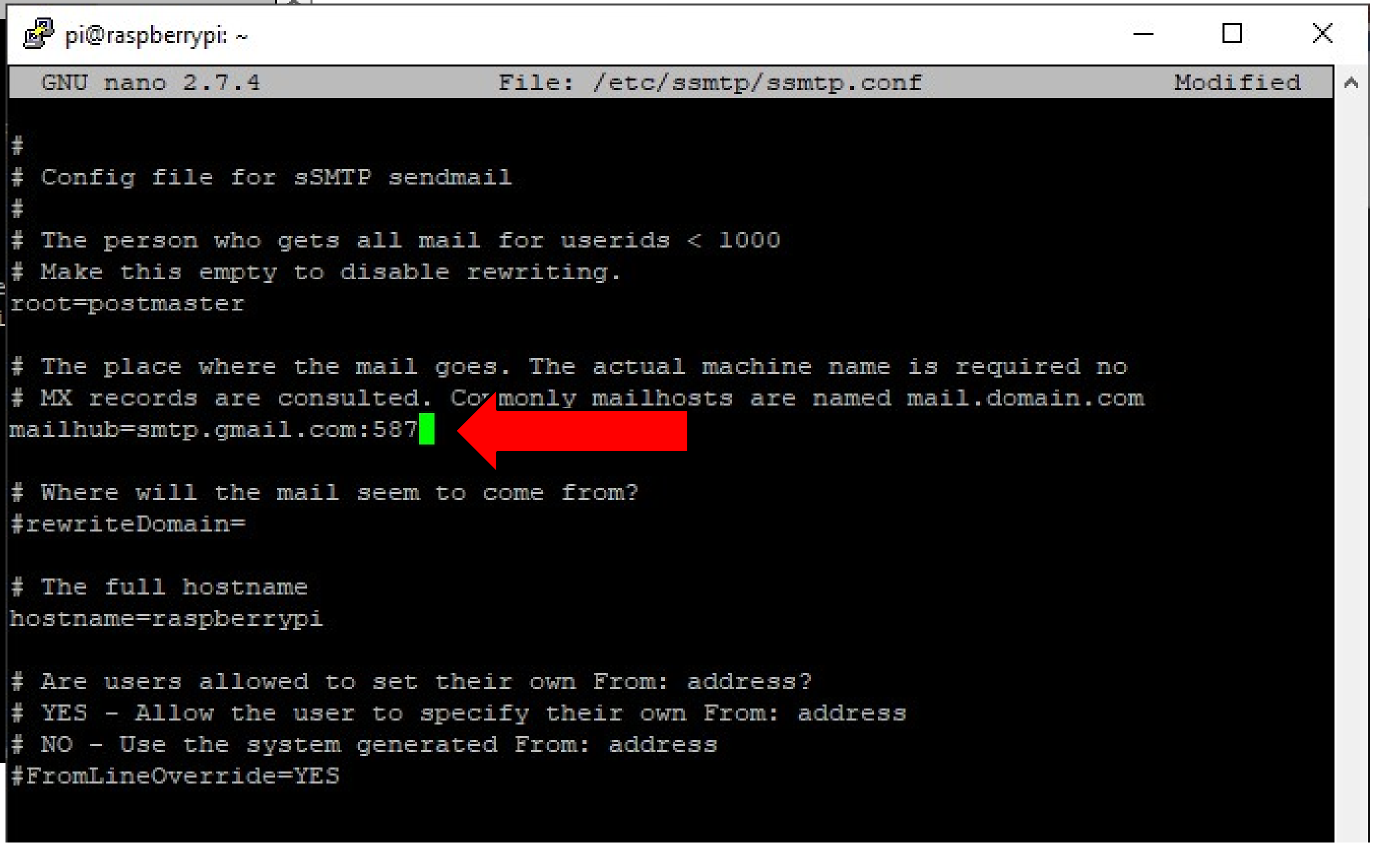
https://www.youtube.com/watch?time_continue=13&v=0kpGcMjpDcw

Localization: Communications

- Update the mailhub to **mailhub=smtp.gmail.com:587**



```
pi@raspberrypi: ~
  GNU nano 2.7.4              File: /etc/ssmtp/ssmtp.conf

#
# Config file for sSMTP sendmail
#
# The person who gets all mail for userids < 1000
# Make this empty to disable rewriting.
root=postmaster

# The place where the mail goes. The actual machine name is require
# MX records ar  consulted. Commonly mailhosts are named mail.domai
mailhub=mail

# Where will the mail seem to come from?
#rewriteDomain=

# The full hostname
hostname=raspberrypi

# Are users allowed to set their own From: address?
# YES - Allow the user to specify their own From: address
# NO - Use the system generated From: address
#FromLineOverride=YES
```

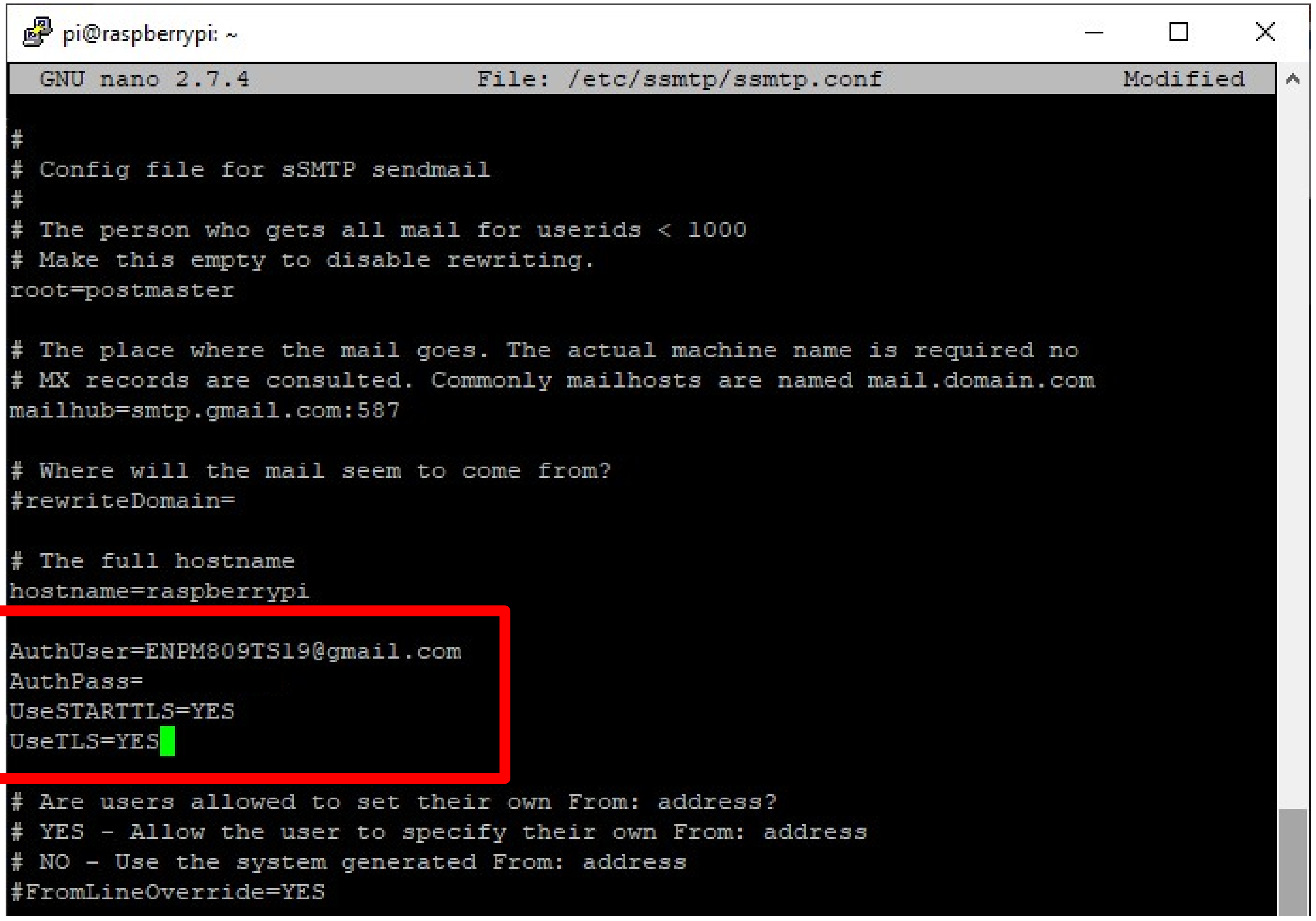


```
pi@raspberrypi: ~
  GNU nano 2.7.4              File: /etc/ssmtp/ssmtp.conf              Modified

#
# Config file for sSMTP sendmail
#
# The person who gets all mail for userids < 1000
# Make this empty to disable rewriting.
root=postmaster

# The place where the mail goes. The actual machine name is required no
# MX records are consulted. Co monly mailhosts are named mail.domain.com
mailhub=smtp.gmail.com:587

# Where will the mail seem to come from?
#rewriteDomain=

# The full hostname
hostname=raspberrypi

# Are users allowed to set their own From: address?
# YES - Allow the user to specify their own From: address
# NO - Use the system generated From: address
#FromLineOverride=YES
```

https://www.youtube.com/watch?time_continue=13&v=0kpGcMjpDcw

Localization: Communications

- Add username, password, & authentication information



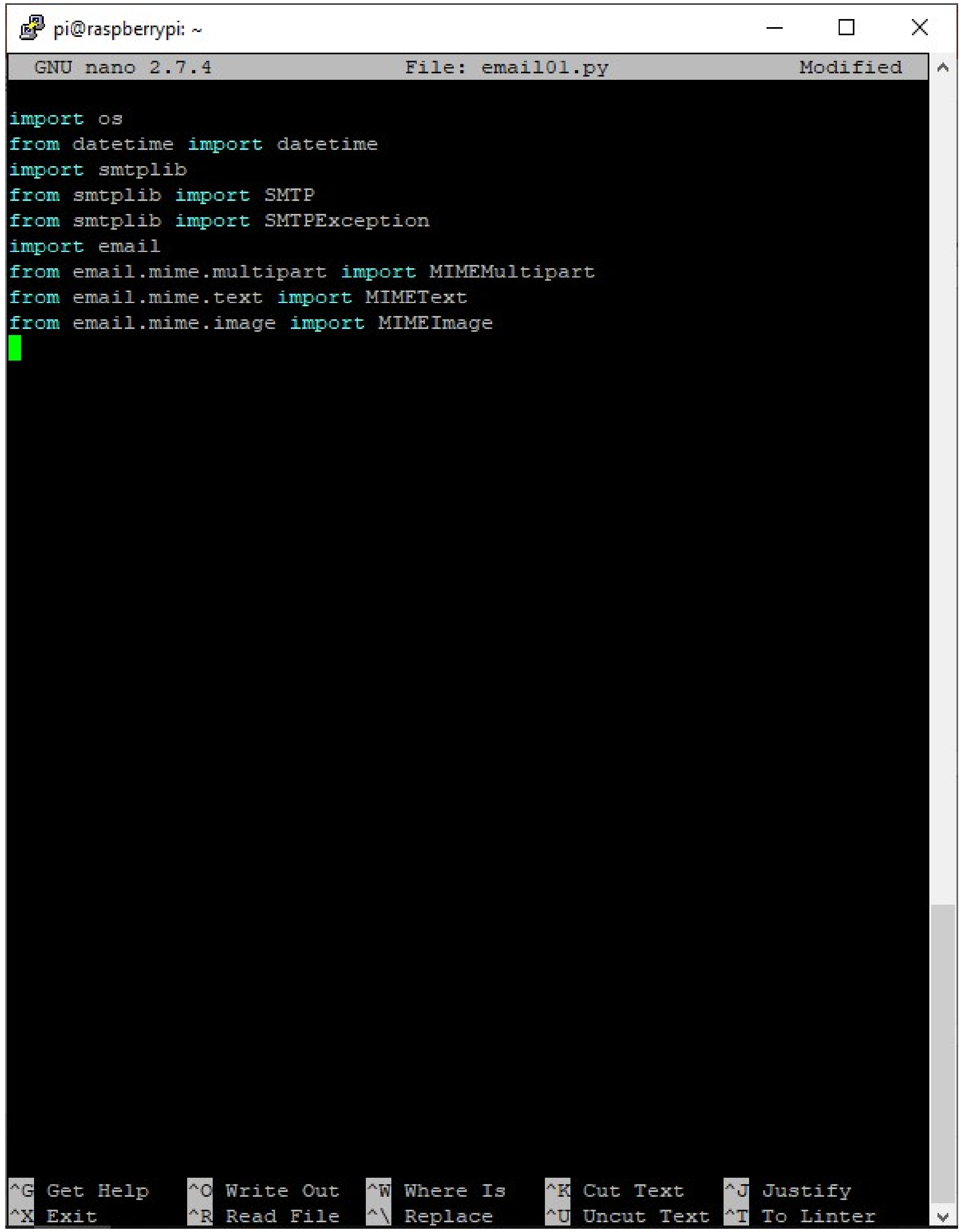
pi@raspberrypi: ~

```
  GNU nano 2.7.4                   File: /etc/ssmtp/ssmtp.conf                Modified

#
# Config file for sSMTP sendmail
#
# The person who gets all mail for userids < 1000
# Make this empty to disable rewriting.
root=postmaster

# The place where the mail goes. The actual machine name is required no
# MX records are consulted. Commonly mailhosts are named mail.domain.com
mailhub=smtp.gmail.com:587

# Where will the mail seem to come from?
#rewriteDomain=

# The full hostname
hostname=raspberrypi

AuthUser=ENPM809TS19@gmail.com
AuthPass=
UseSTARTTLS=YES
UseTLS=YES

# Are users allowed to set their own From: address?
# YES - Allow the user to specify their own From: address
# NO - Use the system generated From: address
#FromLineOverride=YES
```

https://www.youtube.com/watch?time_continue=13&v=0kpGcMjpDcw

Localization

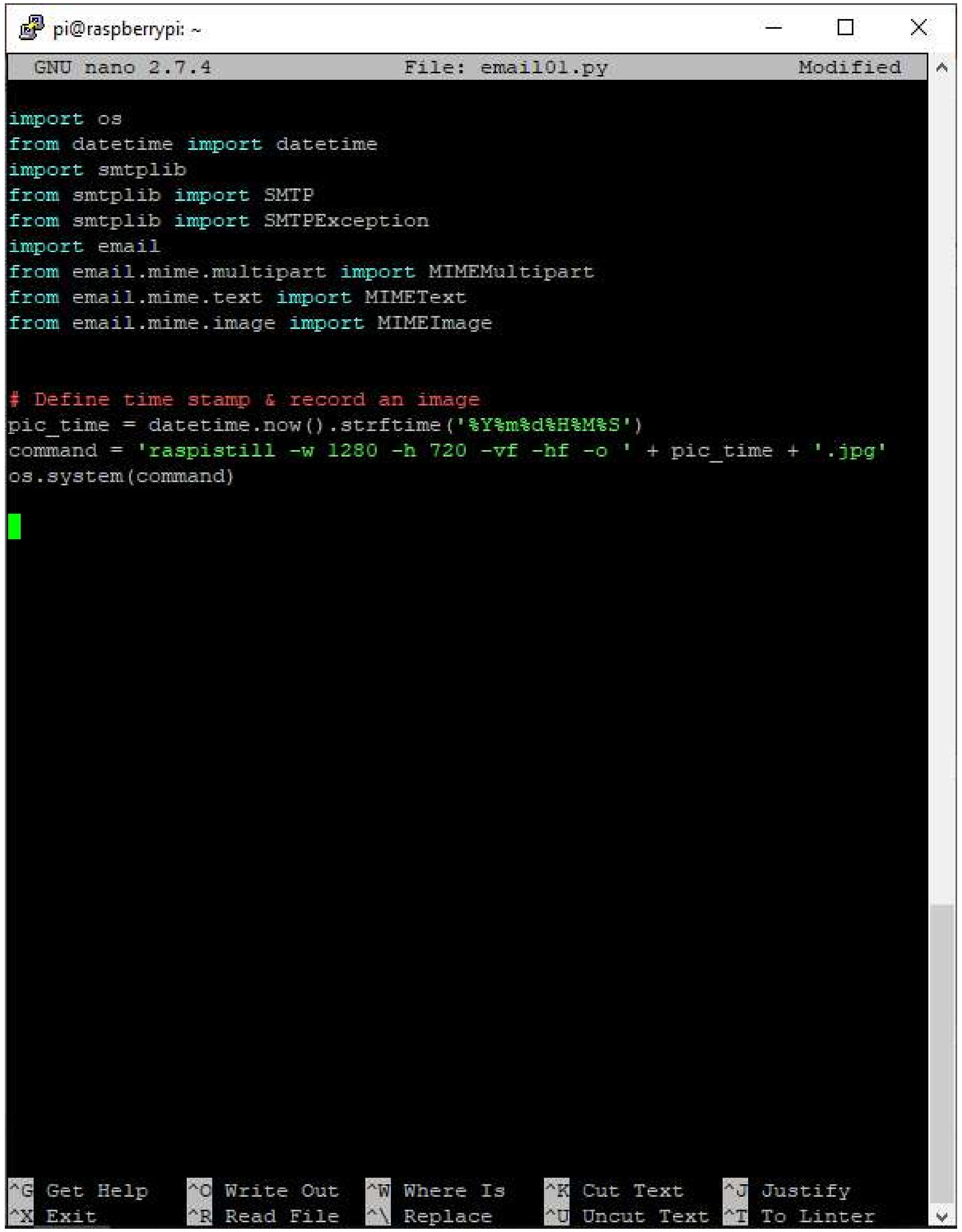
- Create a new Python script: ***email01.py***
- Import required packages



```
  GNU nano 2.7.4                    File: email01.py                    Modified

import os
from datetime import datetime
import smtplib
from smtplib import SMTP
from smtplib import SMTPException
import email
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Linter
```

Localization

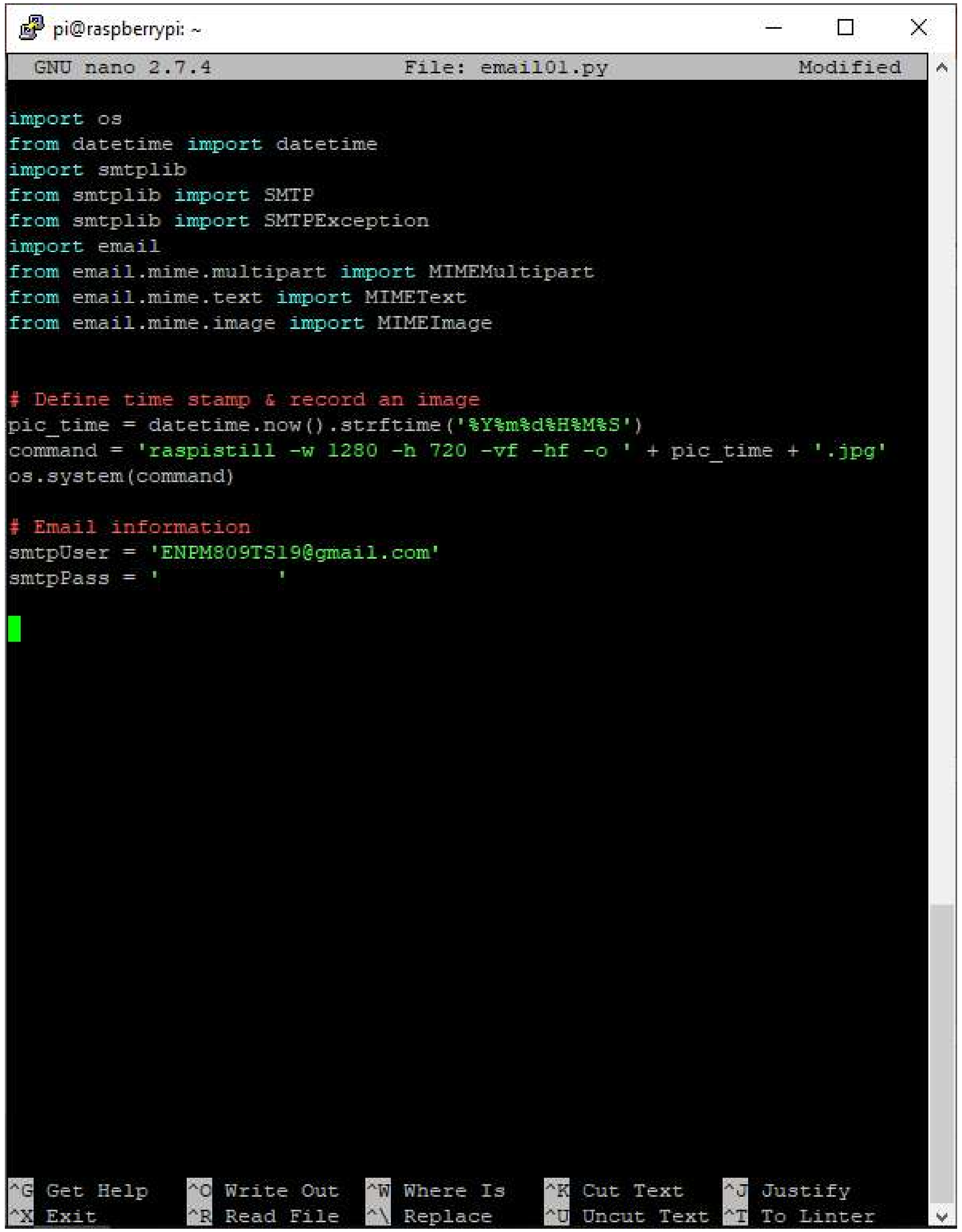
- Create unique time stamp
- Record single image using raspistill & os.system( ) command



```
pi@raspberrypi: ~
  GNU nano 2.7.4                File: email01.py                 Modified

import os
from datetime import datetime
import smtplib
from smtplib import SMTP
from smtplib import SMTPException
import email
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage


# Define time stamp & record an image
pic_time = datetime.now().strftime('%Y%m%d%H%M%S')
command = 'raspistill -w 1280 -h 720 -vf -hf -o ' + pic_time + '.jpg'
os.system(command)
```

Localization

- Enter username & password of outgoing mail server (i.e. *your* user & psswd)

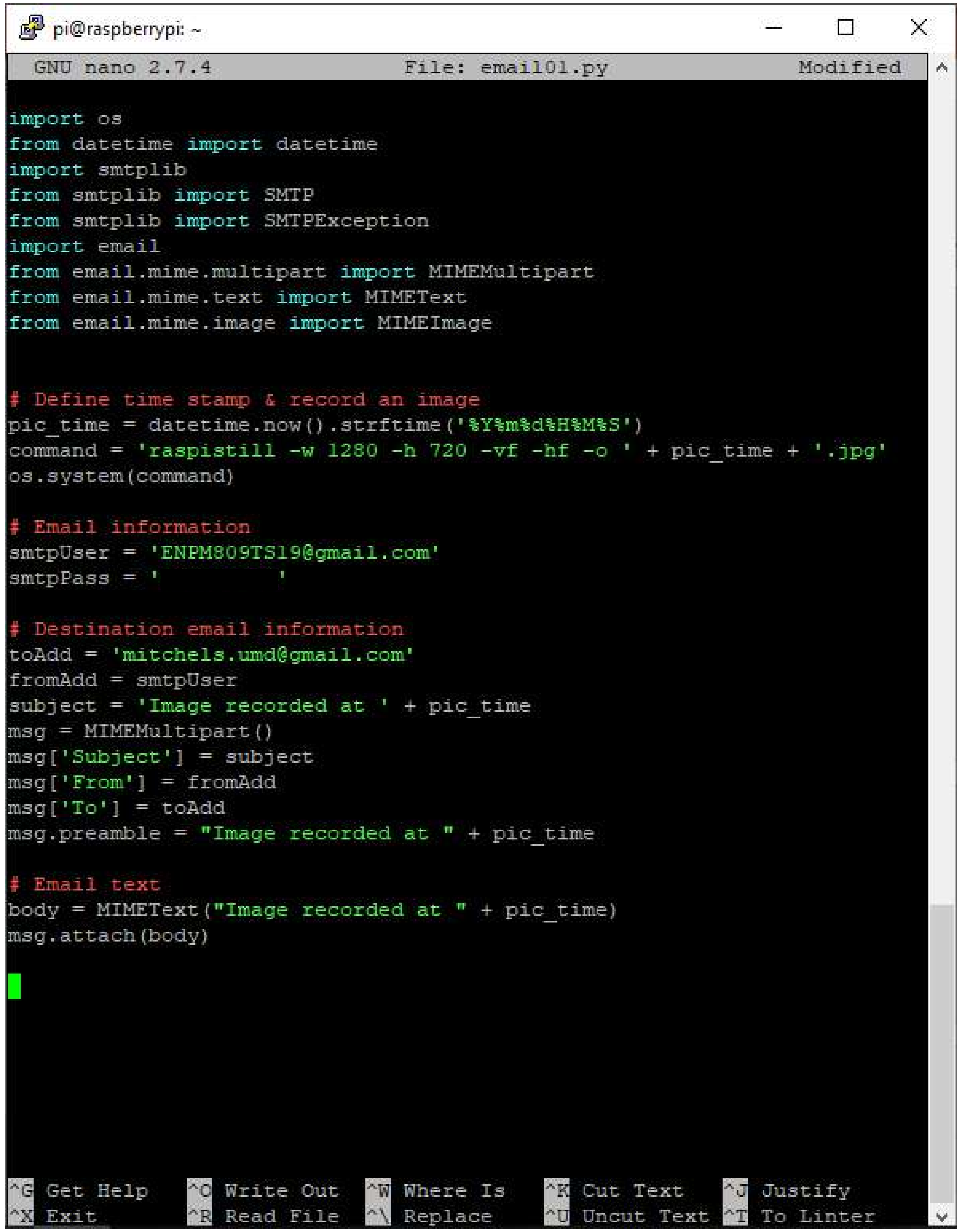


pi@raspberrypi: ~

```
  GNU nano 2.7.4                  File: email01.py                   Modified

import os
from datetime import datetime
import smtplib
from smtplib import SMTP
from smtplib import SMTPException
import email
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage


# Define time stamp & record an image
pic_time = datetime.now().strftime('%Y%m%d%H%M%S')
command = 'raspistill -w 1280 -h 720 -vf -hf -o ' + pic_time + '.jpg'
os.system(command)

# Email information
smtpUser = 'ENPM809TS19@gmail.com'
smtpPass = '         '


^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Linter
```

Localization

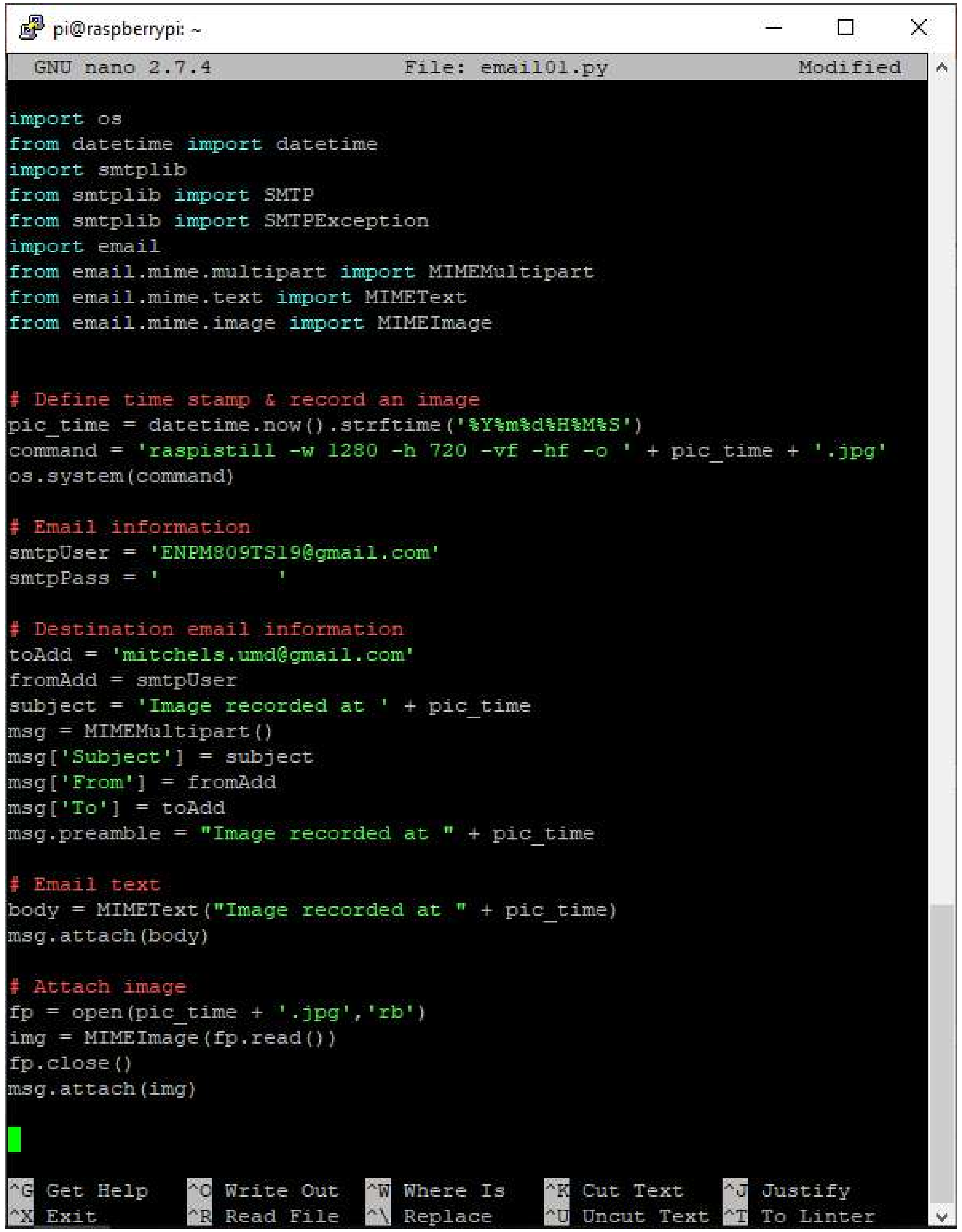
- Enter destination email information
- Enter body of email



```
  GNU nano 2.7.4                File: email01.py                Modified

import os
from datetime import datetime
import smtplib
from smtplib import SMTP
from smtplib import SMTPException
import email
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage


# Define time stamp & record an image
pic_time = datetime.now().strftime('%Y%m%d%H%M%S')
command = 'raspistill -w 1280 -h 720 -vf -hf -o ' + pic_time + '.jpg'
os.system(command)

# Email information
smtpUser = 'ENPM809TS19@gmail.com'
smtpPass = '         '

# Destination email information
toAdd = 'mitchels.umd@gmail.com'
fromAdd = smtpUser
subject = 'Image recorded at ' + pic_time
msg = MIMEMultipart()
msg['Subject'] = subject
msg['From'] = fromAdd
msg['To'] = toAdd
msg.preamble = "Image recorded at " + pic_time

# Email text
body = MIMEText("Image recorded at " + pic_time)
msg.attach(body)



^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Linter
```

Localization

- Attach image recorded using raspistill to email

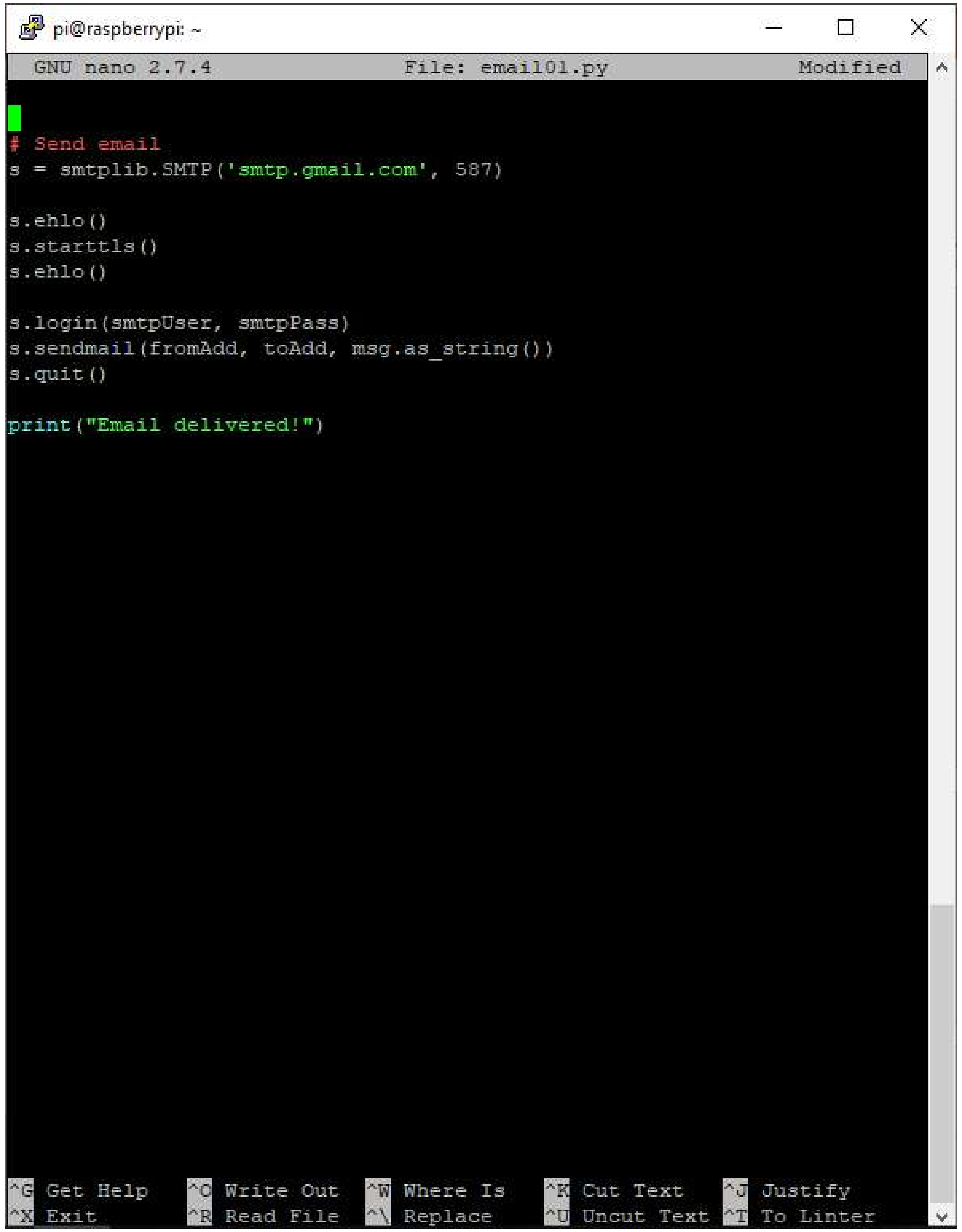


pi@raspberrypi: ~

```
  GNU nano 2.7.4                    File: email01.py                     Modified

import os
from datetime import datetime
import smtplib
from smtplib import SMTP
from smtplib import SMTPException
import email
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage


# Define time stamp & record an image
pic_time = datetime.now().strftime('%Y%m%d%H%M%S')
command = 'raspistill -w 1280 -h 720 -vf -hf -o ' + pic_time + '.jpg'
os.system(command)

# Email information
smtpUser = 'ENPM809TS19@gmail.com'
smtpPass = '          '

# Destination email information
toAdd = 'mitchels.umd@gmail.com'
fromAdd = smtpUser
subject = 'Image recorded at ' + pic_time
msg = MIMEMultipart()
msg['Subject'] = subject
msg['From'] = fromAdd
msg['To'] = toAdd
msg.preamble = "Image recorded at " + pic_time

# Email text
body = MIMEText("Image recorded at " + pic_time)
msg.attach(body)

# Attach image
fp = open(pic_time + '.jpg','rb')
img = MIMEImage(fp.read())
fp.close()
msg.attach(img)


^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Linter
```

Localization

- Finally, deliver email to destination

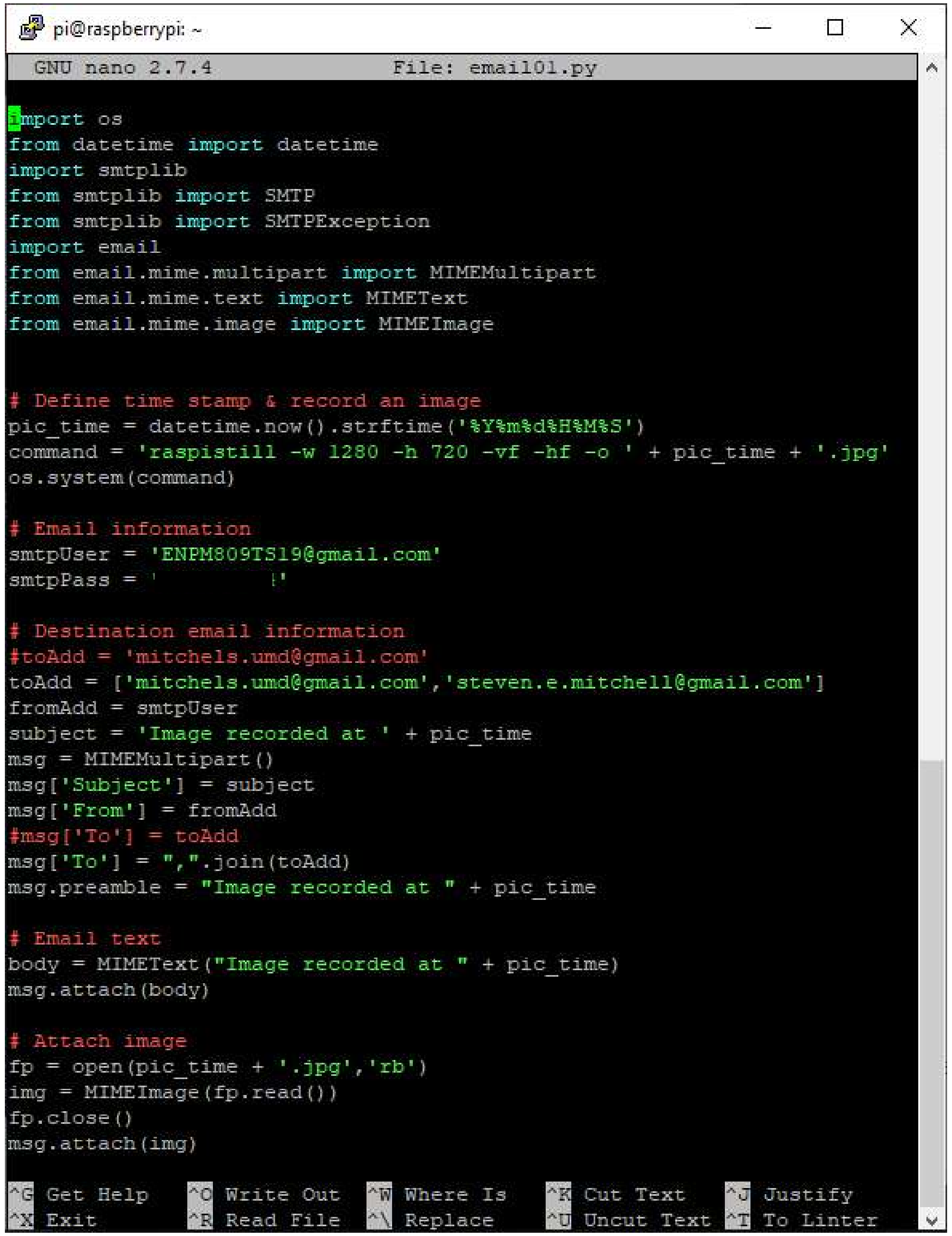


pi@raspberrypi: ~

```
  GNU nano 2.7.4                File: email01.py                Modified


# Send email
s = smtplib.SMTP('smtp.gmail.com', 587)

s.ehlo()
s.starttls()
s.ehlo()

s.login(smtpUser, smtpPass)
s.sendmail(fromAdd, toAdd, msg.as_string())
s.quit()

print("Email delivered!")
```

Localization

- To deliver email to multiple users:



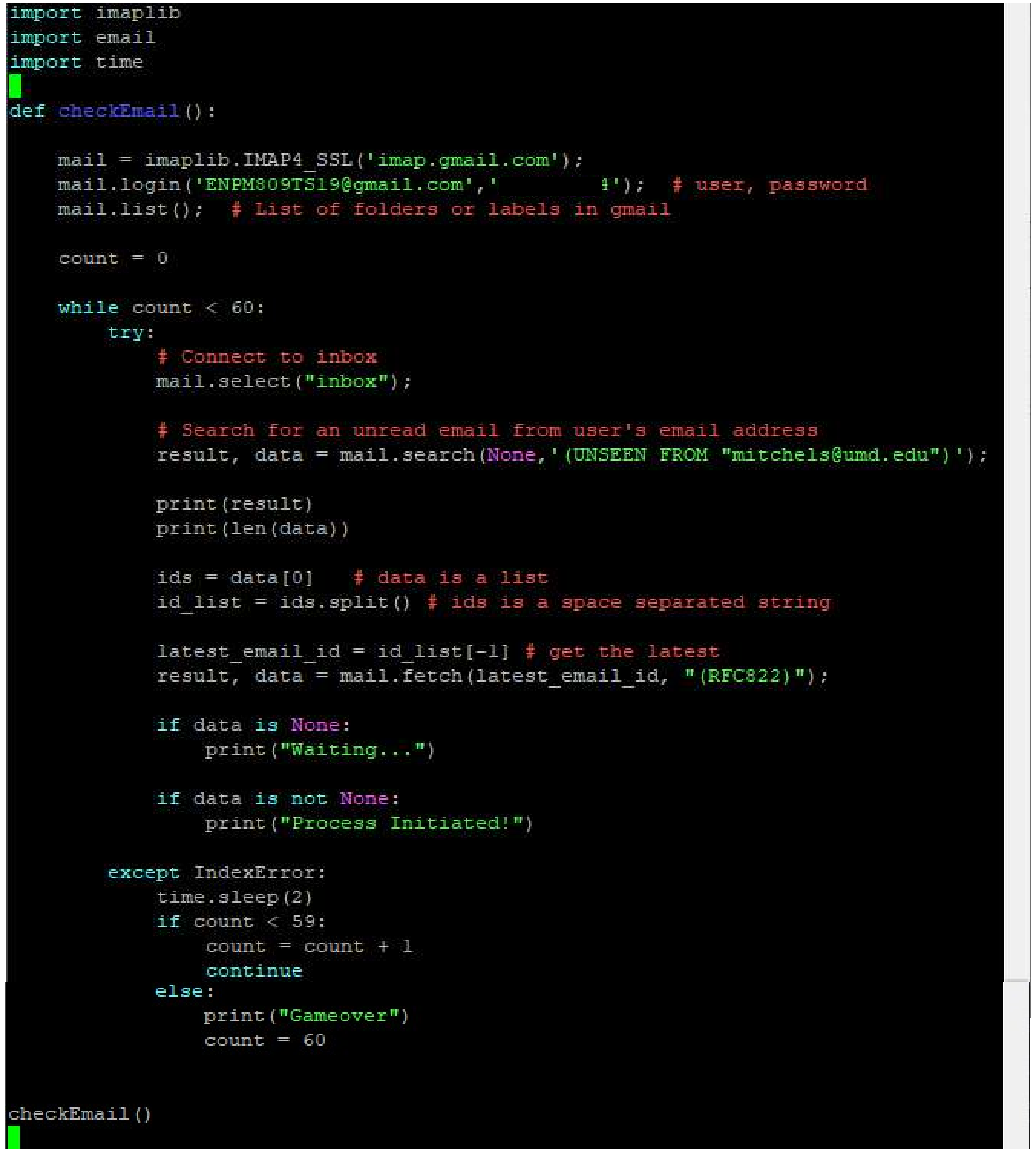
```
  GNU nano 2.7.4                File: email01.py

import os
from datetime import datetime
import smtplib
from smtplib import SMTP
from smtplib import SMTPException
import email
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage


# Define time stamp & record an image
pic_time = datetime.now().strftime('%Y%m%d%H%M%S')
command = 'raspistill -w 1280 -h 720 -vf -hf -o ' + pic_time + '.jpg'
os.system(command)

# Email information
smtpUser = 'ENPM809TS19@gmail.com'
smtpPass = '[redacted]!'

# Destination email information
#toAdd = 'mitchels.umd@gmail.com'
toAdd = ['mitchels.umd@gmail.com','steven.e.mitchell@gmail.com']
fromAdd = smtpUser
subject = 'Image recorded at ' + pic_time
msg = MIMEMultipart()
msg['Subject'] = subject
msg['From'] = fromAdd
#msg['To'] = toAdd
msg['To'] = ",".join(toAdd)
msg.preamble = "Image recorded at " + pic_time

# Email text
body = MIMEText("Image recorded at " + pic_time)
msg.attach(body)

# Attach image
fp = open(pic_time + '.jpg','rb')
img = MIMEImage(fp.read())
fp.close()
msg.attach(img)

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Linter
```

In-Class Exercise

- Add camera IoT functionality to robot teleoperation script
- Deliver image via email: **ENPM809TS19@gmail.com**

Localization

- Using email to initiate event on the RPi



```
import imaplib
import email
import time

def checkEmail():

    mail = imaplib.IMAP4_SSL('imap.gmail.com');
    mail.login('ENPM809TS19@gmail.com','[redacted]');  # user, password
    mail.list();  # List of folders or labels in gmail

    count = 0

    while count < 60:
        try:
            # Connect to inbox
            mail.select("inbox");

            # Search for an unread email from user's email address
            result, data = mail.search(None,'(UNSEEN FROM "mitchels@umd.edu")');

            print(result)
            print(len(data))

            ids = data[0]   # data is a list
            id_list = ids.split() # ids is a space separated string

            latest_email_id = id_list[-1] # get the latest
            result, data = mail.fetch(latest_email_id, "(RFC822)");

            if data is None:
                print("Waiting...")

            if data is not None:
                print("Process Initiated!")

        except IndexError:
            time.sleep(2)
            if count < 59:
                count = count + 1
                continue
            else:
                print("Gameover")
                count = 60


checkEmail()
```

In-Class Exercise

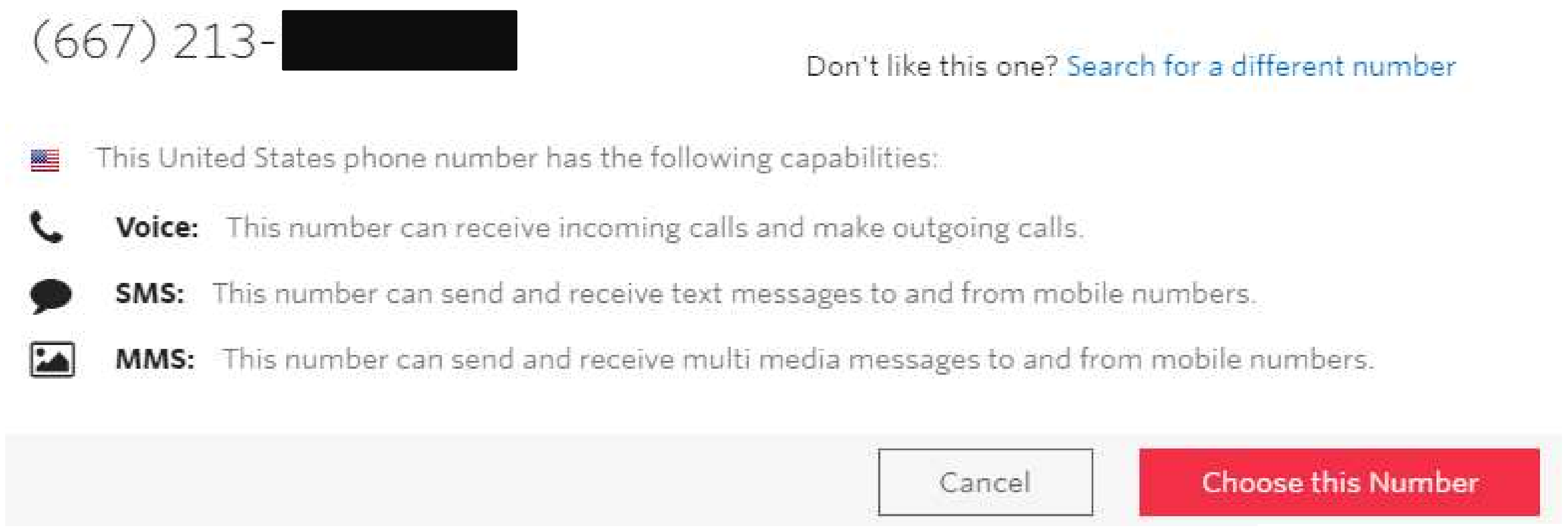
- Demonstrate ability to initiate event on RPi via email ping
- Initiation email will be sent from: **ENPM809TS19@gmail.com**

Twilio

- Cloud communications platform for building messaging applications
- Founded in 2008
- HQ in San Francisco
- Uses Amazon Web Services
- GroupMe uses Twilio's text messaging product to facilitate group chat

https://www.twilio.com/

Twilio



(667) 213-[redacted]

Don't like this one? Search for a different number

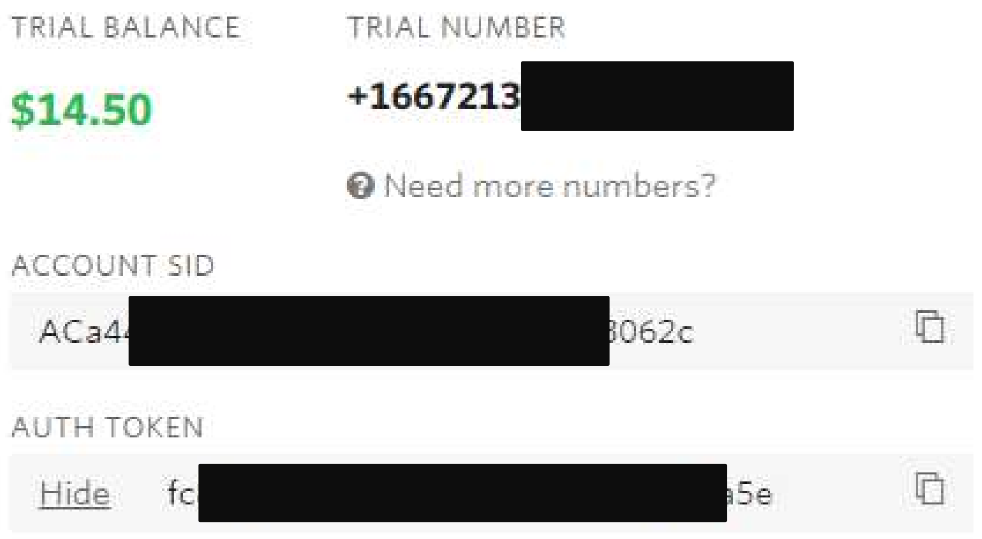
This United States phone number has the following capabilities:

Voice: This number can receive incoming calls and make outgoing calls.

SMS: This number can send and receive text messages to and from mobile numbers.

MMS: This number can send and receive multi media messages to and from mobile numbers.

Cancel | Choose this Number



TRIAL BALANCE

$14.50

TRIAL NUMBER

+1667213[redacted]

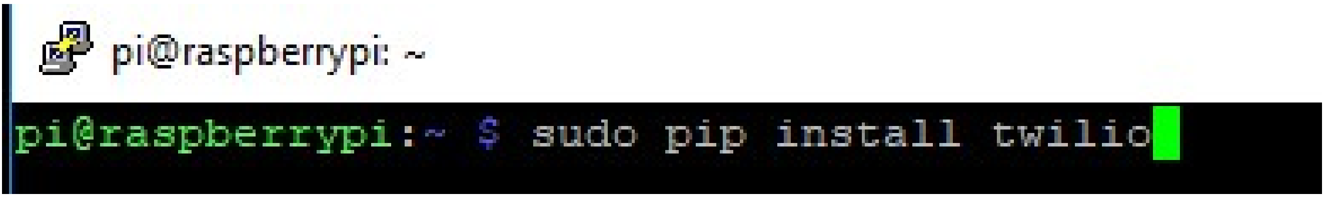
Need more numbers?

ACCOUNT SID

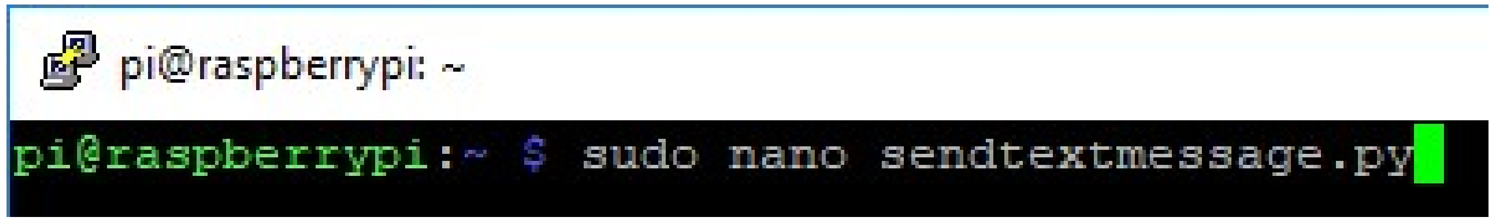
ACa4[redacted]062c

AUTH TOKEN

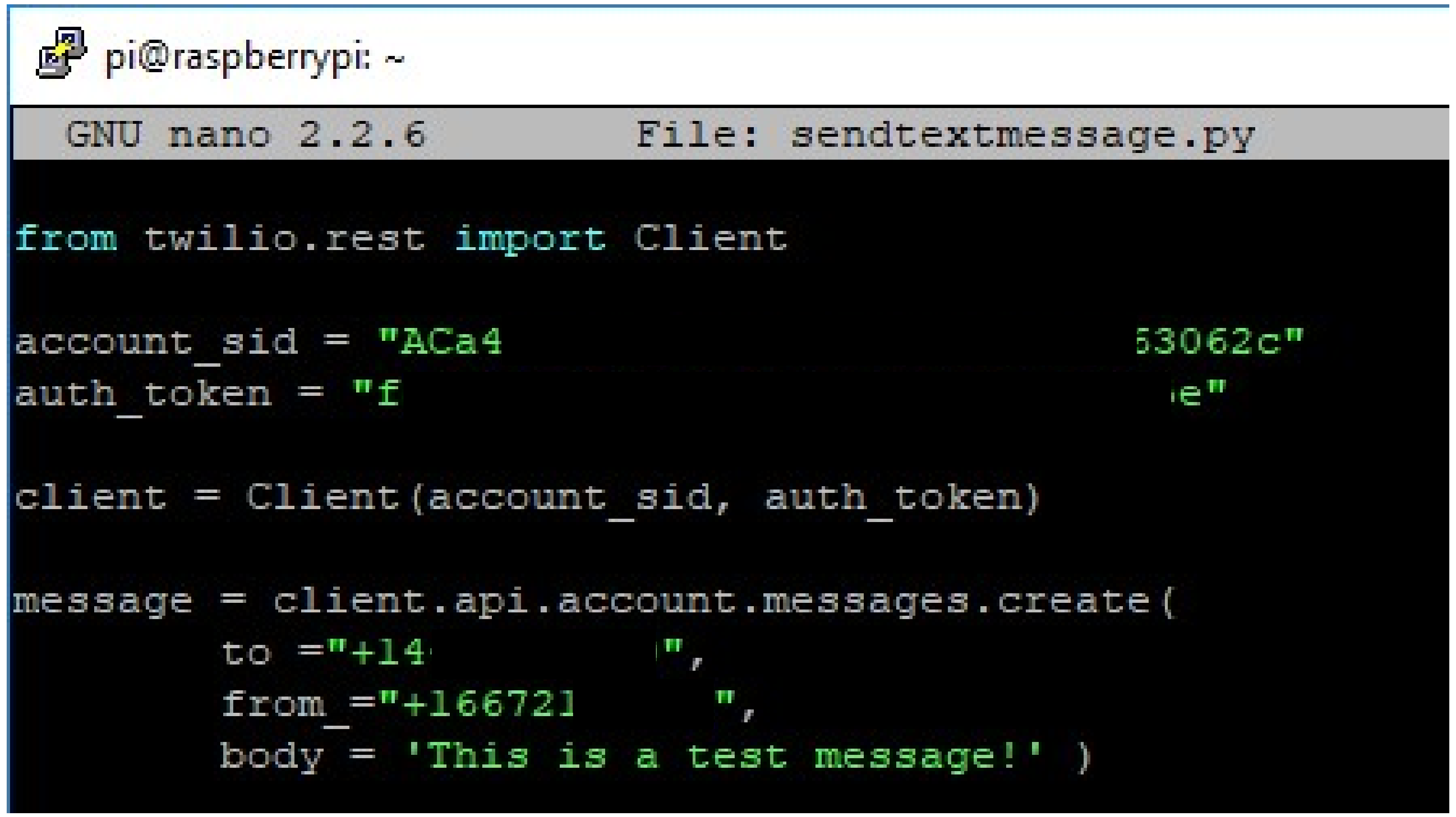
Hide fc[redacted]5e



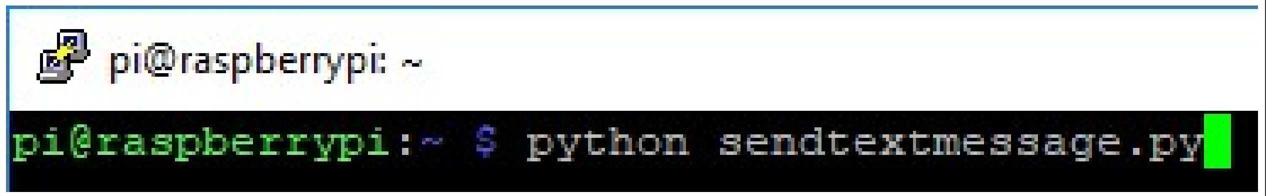
```
pi@raspberrypi:~ $ sudo pip install twilio
```



```
pi@raspberrypi:~ $ sudo nano sendtextmessage.py
```

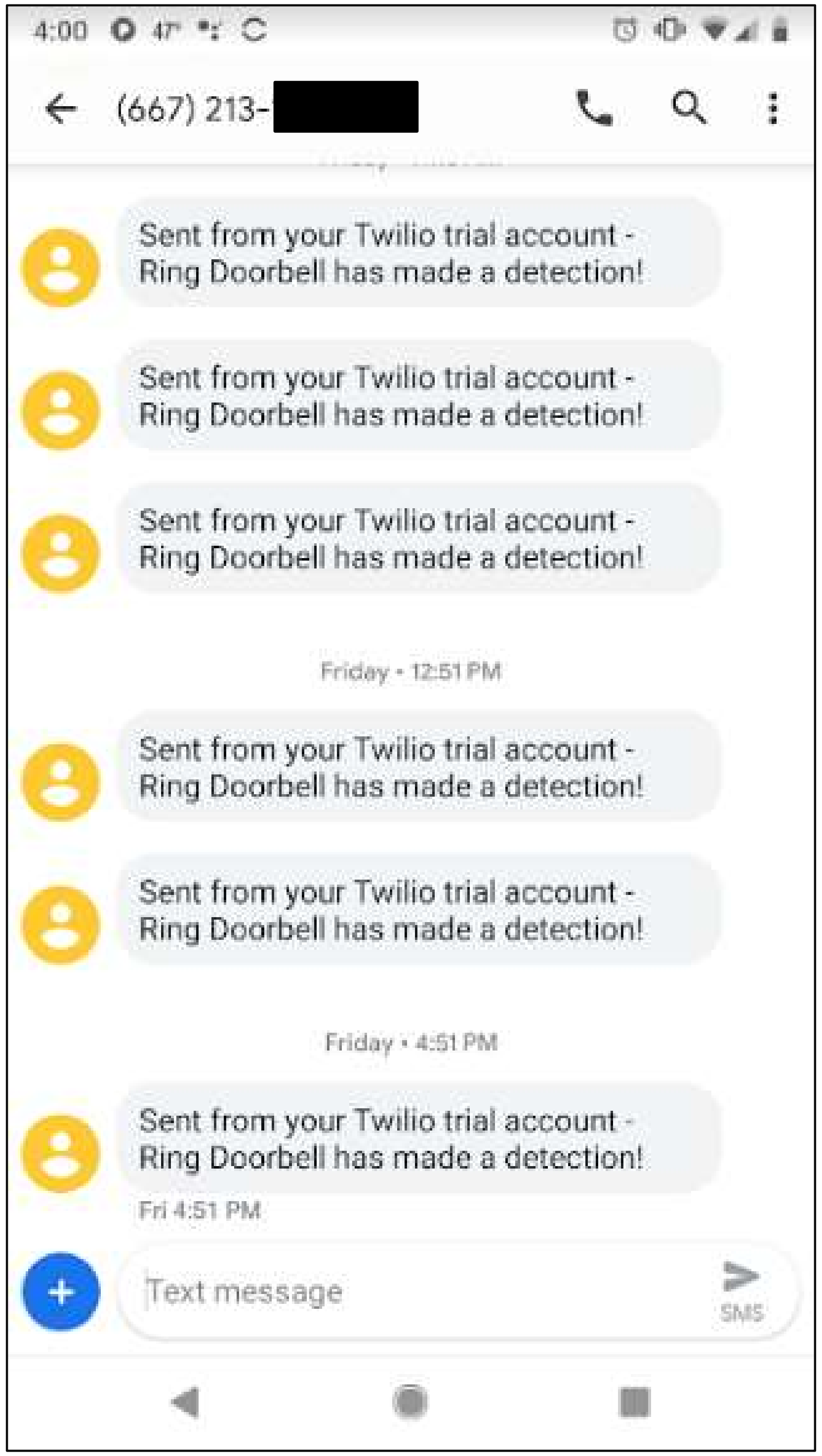


```
  GNU nano 2.2.6          File: sendtextmessage.py

from twilio.rest import Client

account_sid = "ACa4[redacted]53062c"
auth_token = "f[redacted]e"

client = Client(account_sid, auth_token)

message = client.api.account.messages.create(
        to ="+14[redacted]",
        from_="+166721[redacted]",
        body = 'This is a test message!' )
```



```
pi@raspberrypi:~ $ python sendtextmessage.py
```

Twilio


4:00 47°

(667) 213-[redacted]

Sent from your Twilio trial account - Ring Doorbell has made a detection!

Sent from your Twilio trial account - Ring Doorbell has made a detection!

Sent from your Twilio trial account - Ring Doorbell has made a detection!

Friday • 12:51 PM

Sent from your Twilio trial account - Ring Doorbell has made a detection!

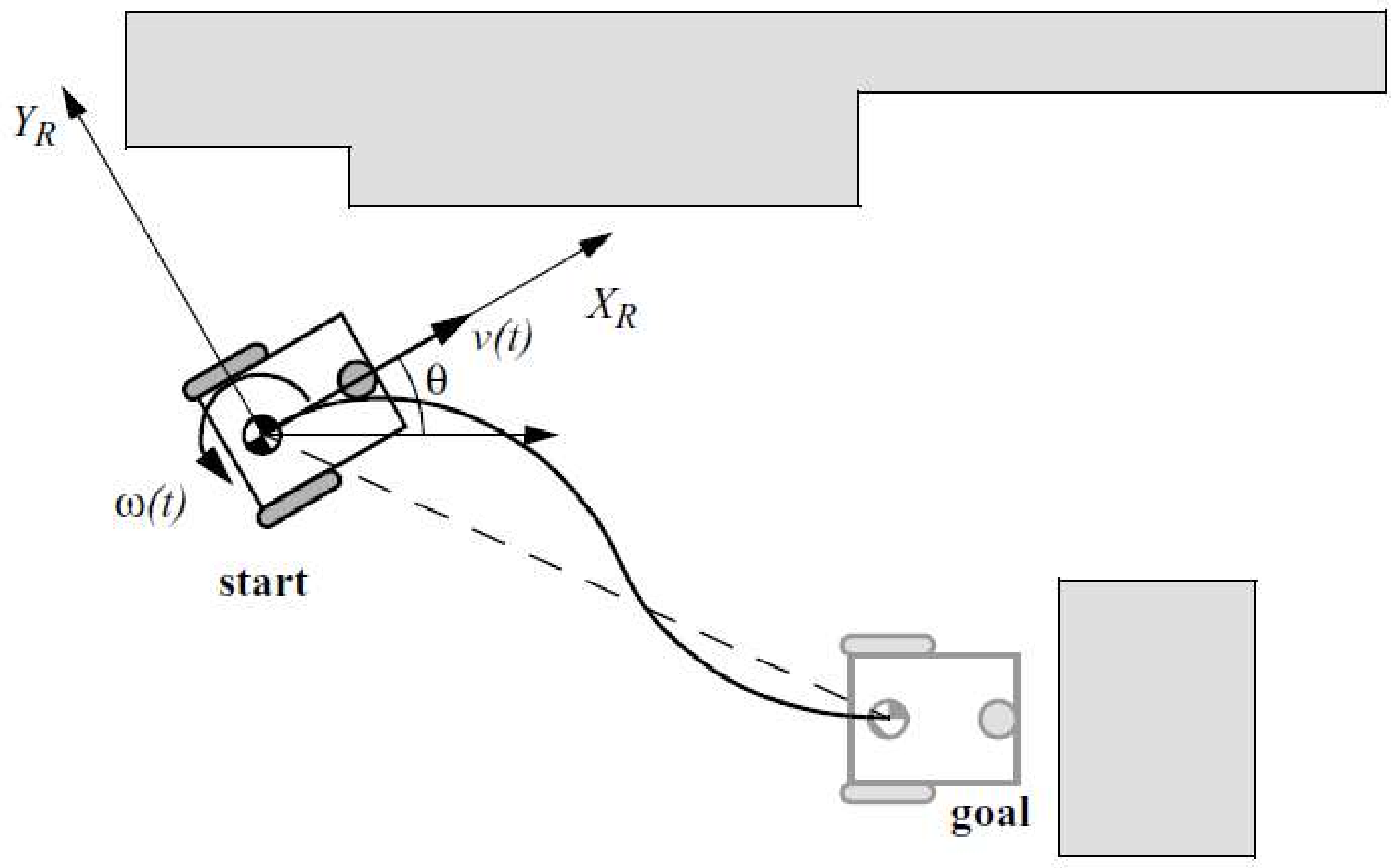
Sent from your Twilio trial account - Ring Doorbell has made a detection!

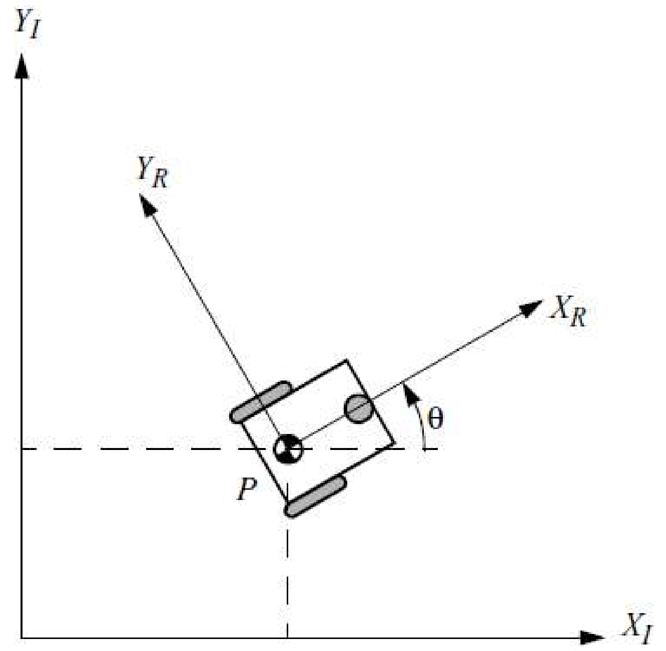
Friday • 4:51 PM

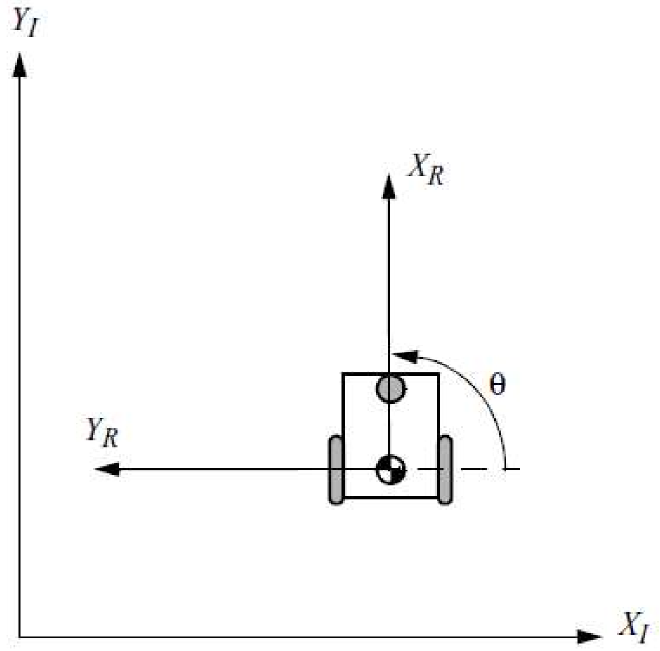
Sent from your Twilio trial account - Ring Doorbell has made a detection!

Fri 4:51 PM

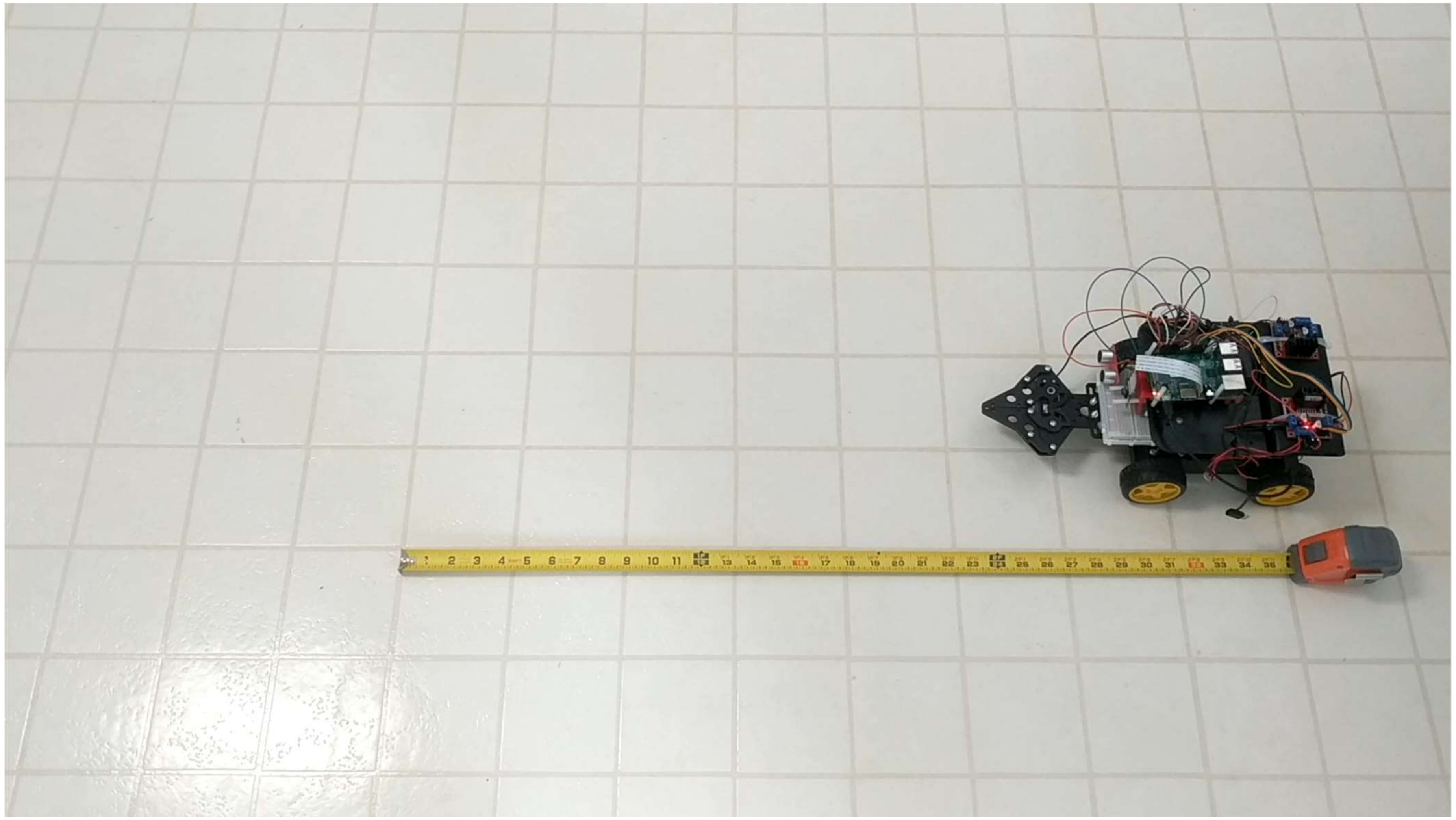
Kinematics & Localization



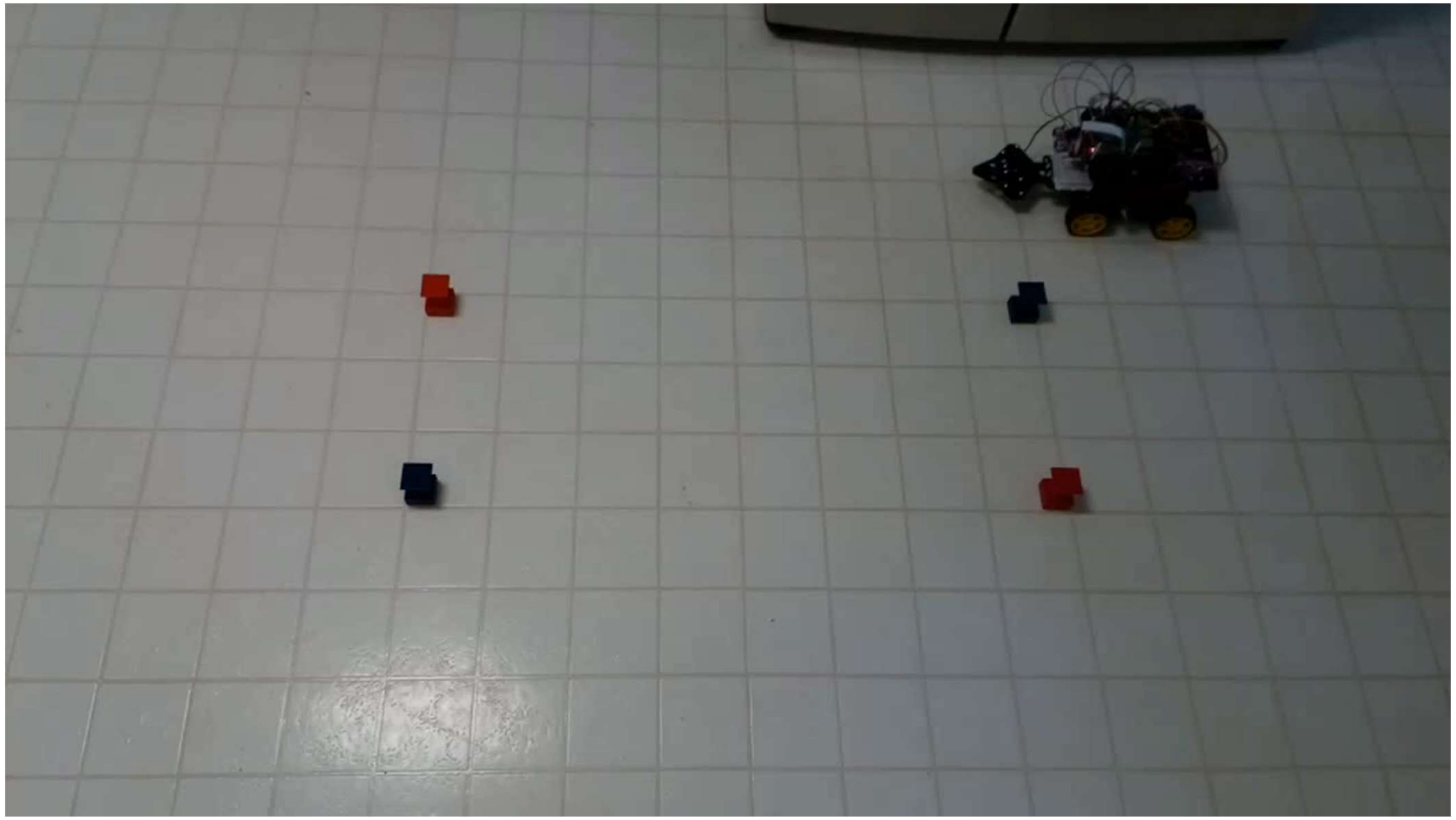




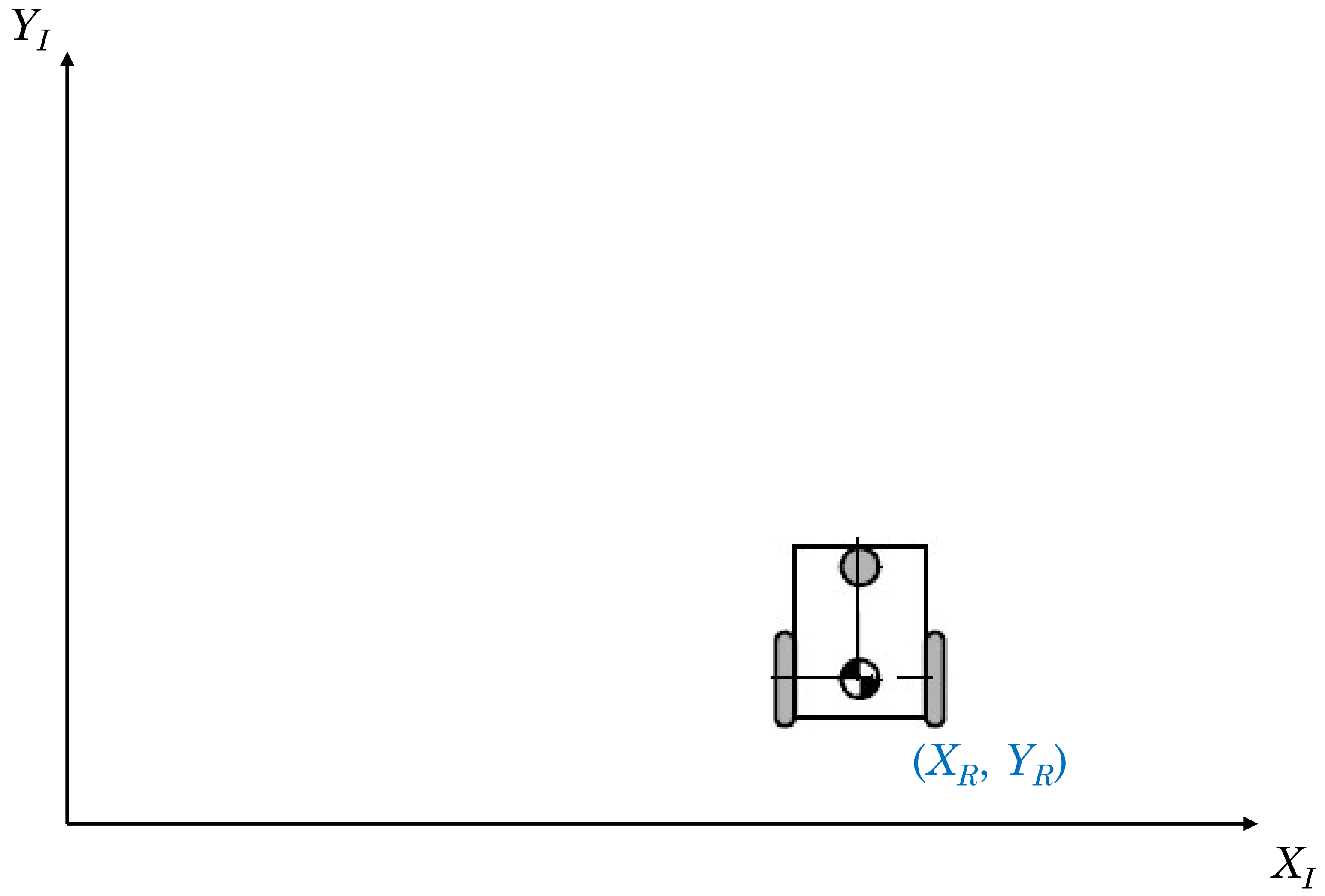
Kinematics & Localization



Kinematics & Localization

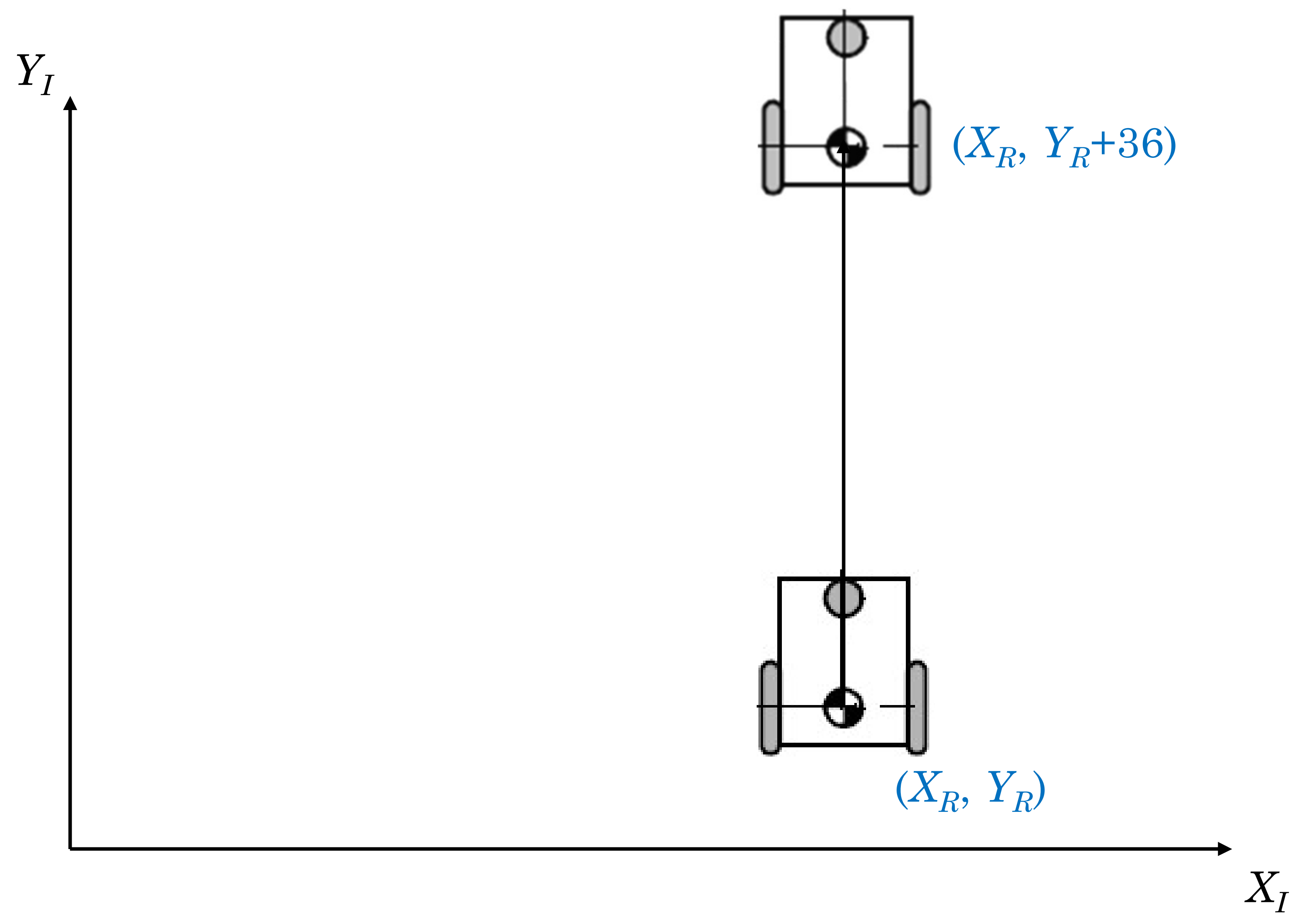


Kinematics & Localization



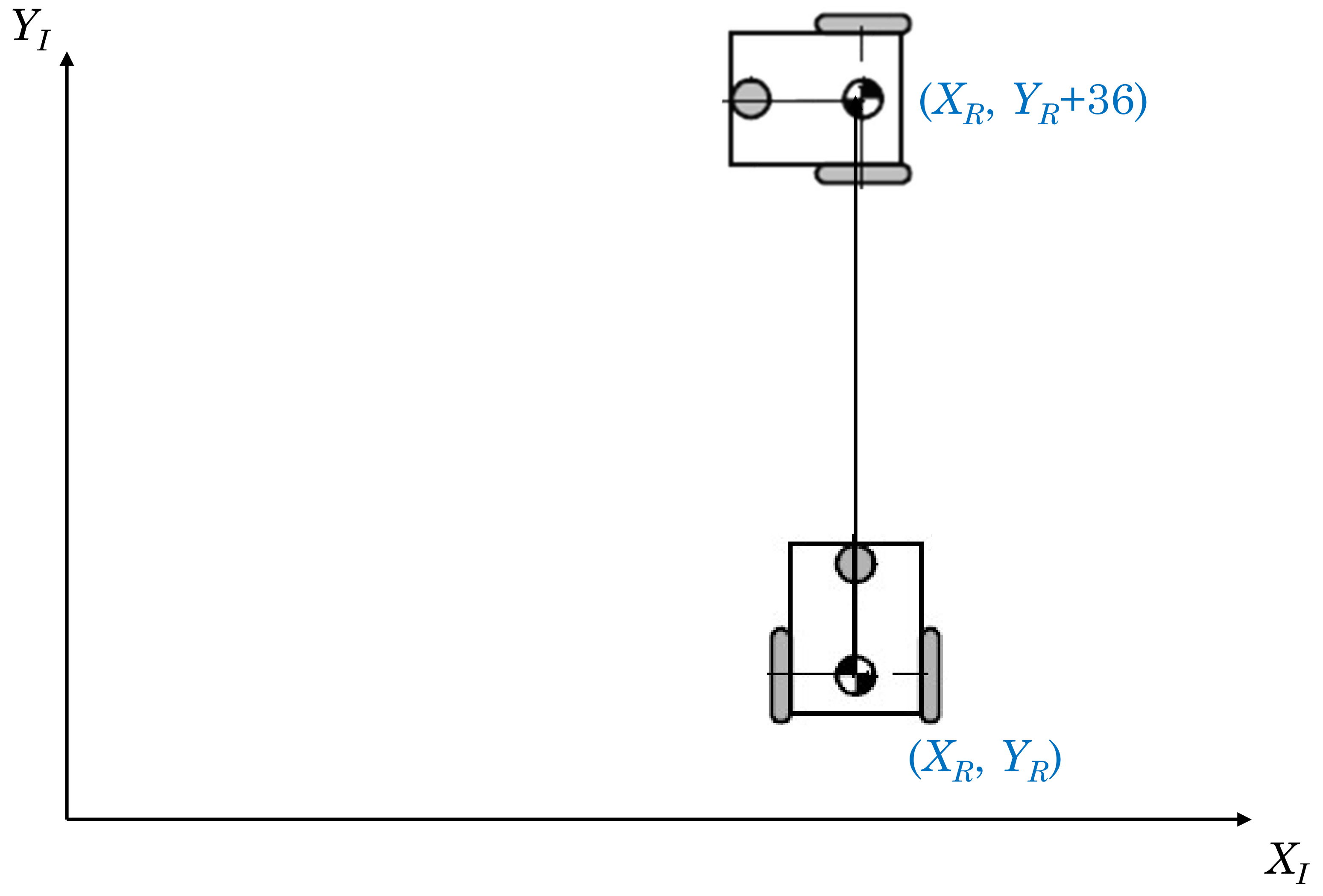
1. Initial location
   - (X_R, Y_R)

Kinematics & Localization



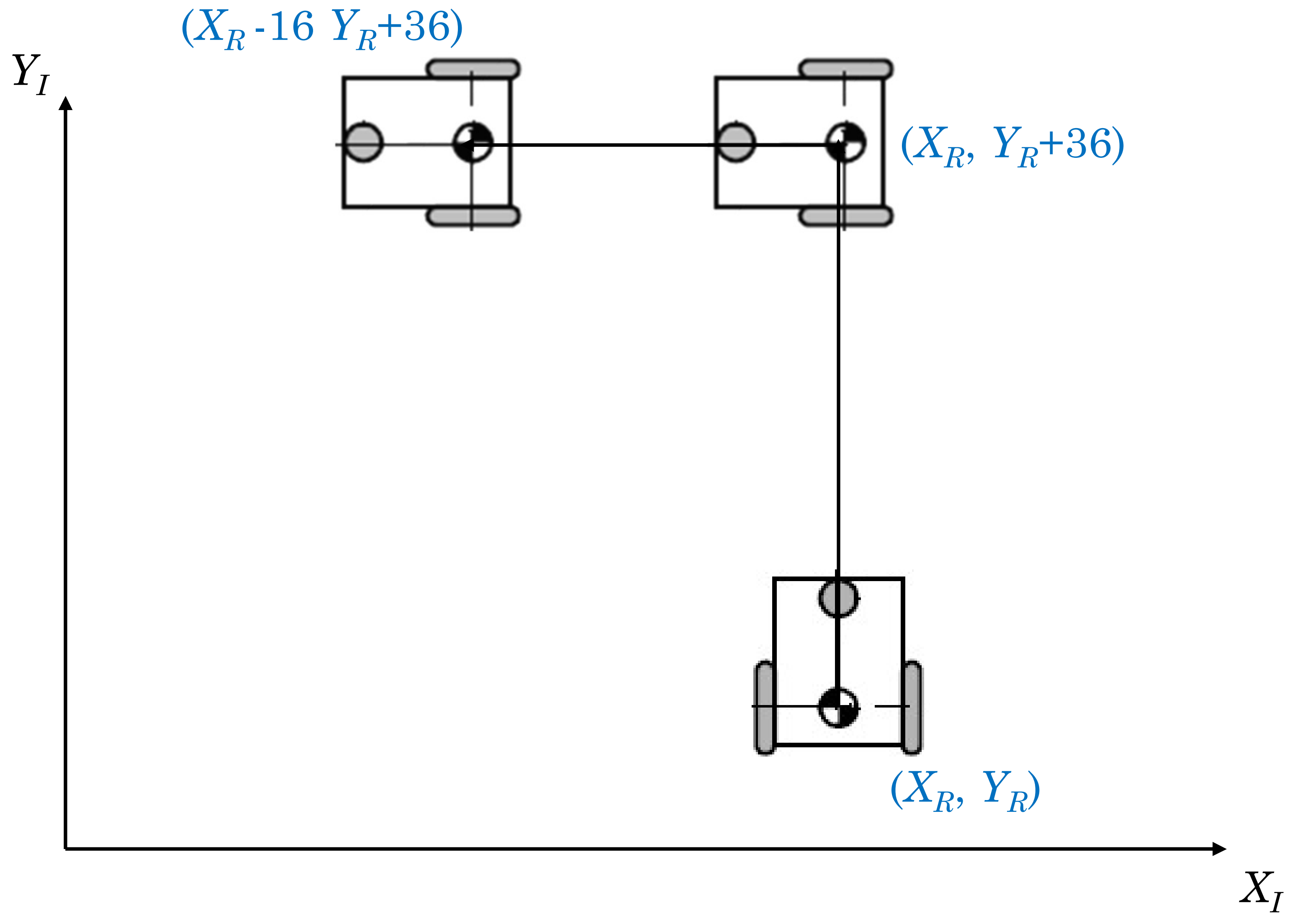
1. Initial location
   - $(X_R,\ Y_R)$
2. Forward 36 in
   - $(X_R,\ Y_R{+}36)$

Kinematics & Localization



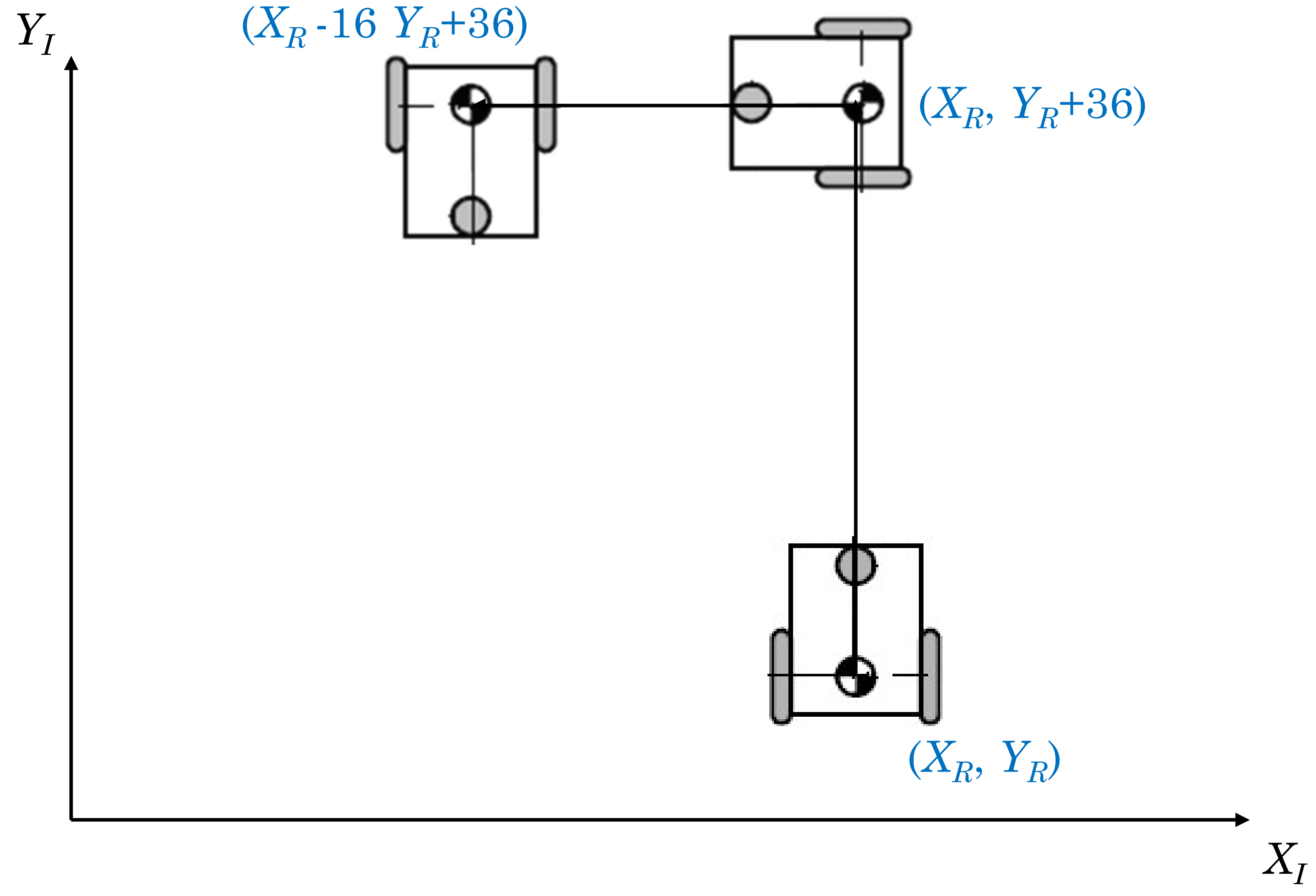
1. Initial location
   - (X_R, Y_R)
2. Forward 36 in
   - (X_R, Y_R+36)
3. Pivot left 90°
   - (X_R, Y_R+36)

Kinematics & Localization



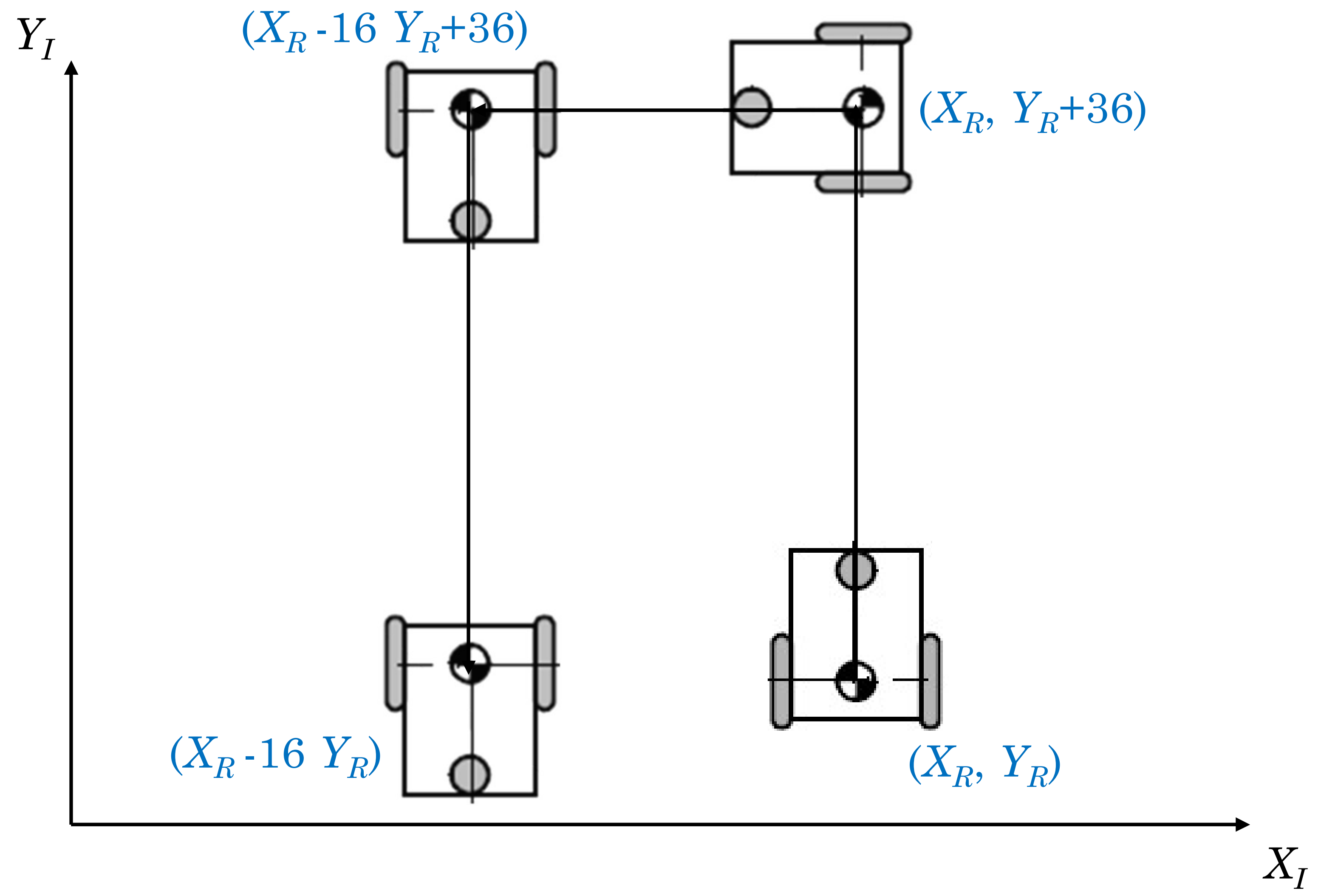
1. Initial location
   - (X_R, Y_R)
2. Forward 36 in
   - (X_R, Y_R+36)
3. Pivot left 90^o
   - (X_R, Y_R+36)
4. Forward 16 in
   - $(X_R - 16, Y_R+36)$

Kinematics & Localization



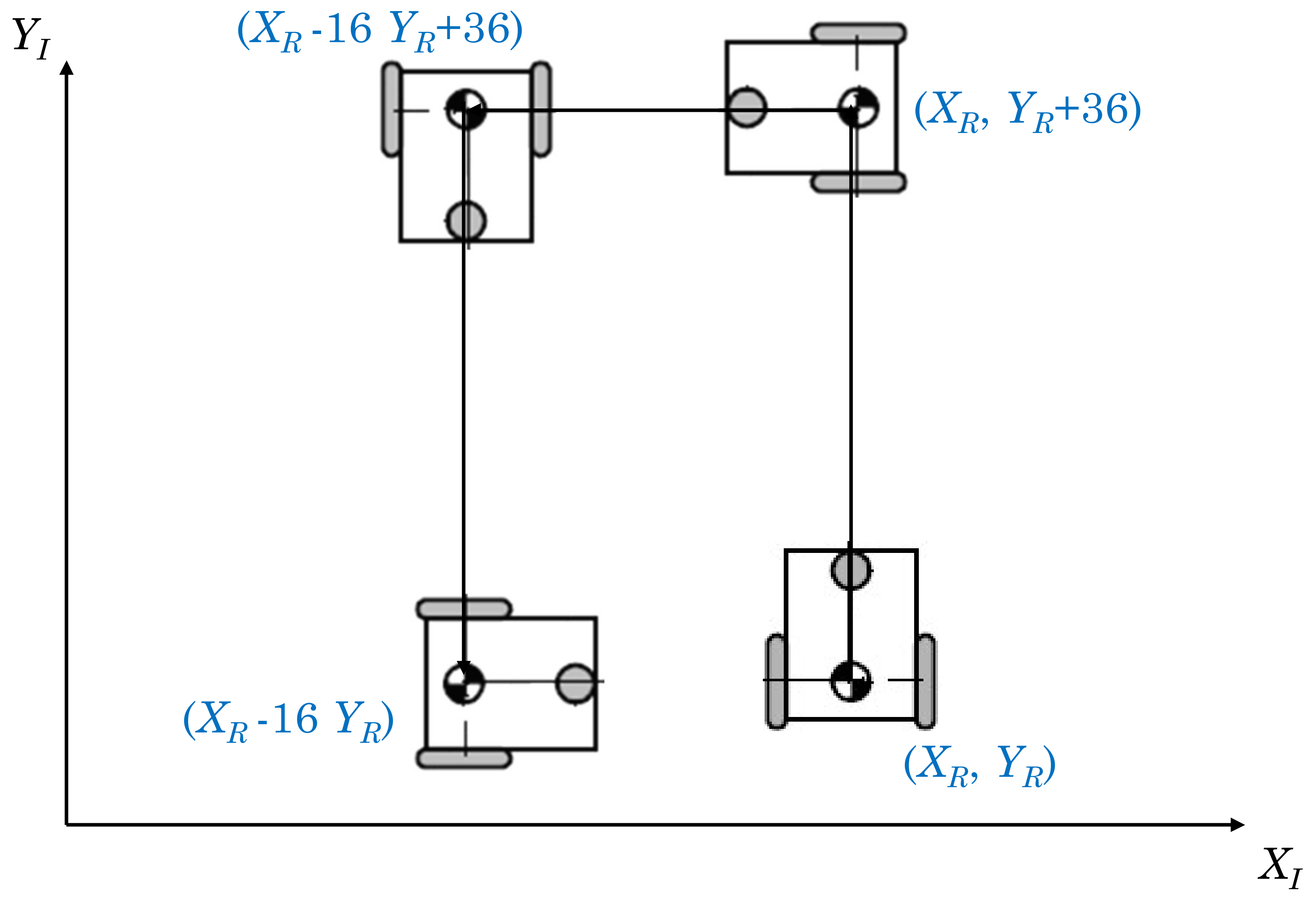
1. Initial location
   - (X_R, Y_R)
2. Forward 36 in
   - (X_R, Y_R+36)
3. Pivot left 90º
   - (X_R, Y_R+36)
4. Forward 16 in
   - (X_R-16, Y_R+36)
5. Pivot left 90º
   - (X_R-16, Y_R+36)

Kinematics & Localization



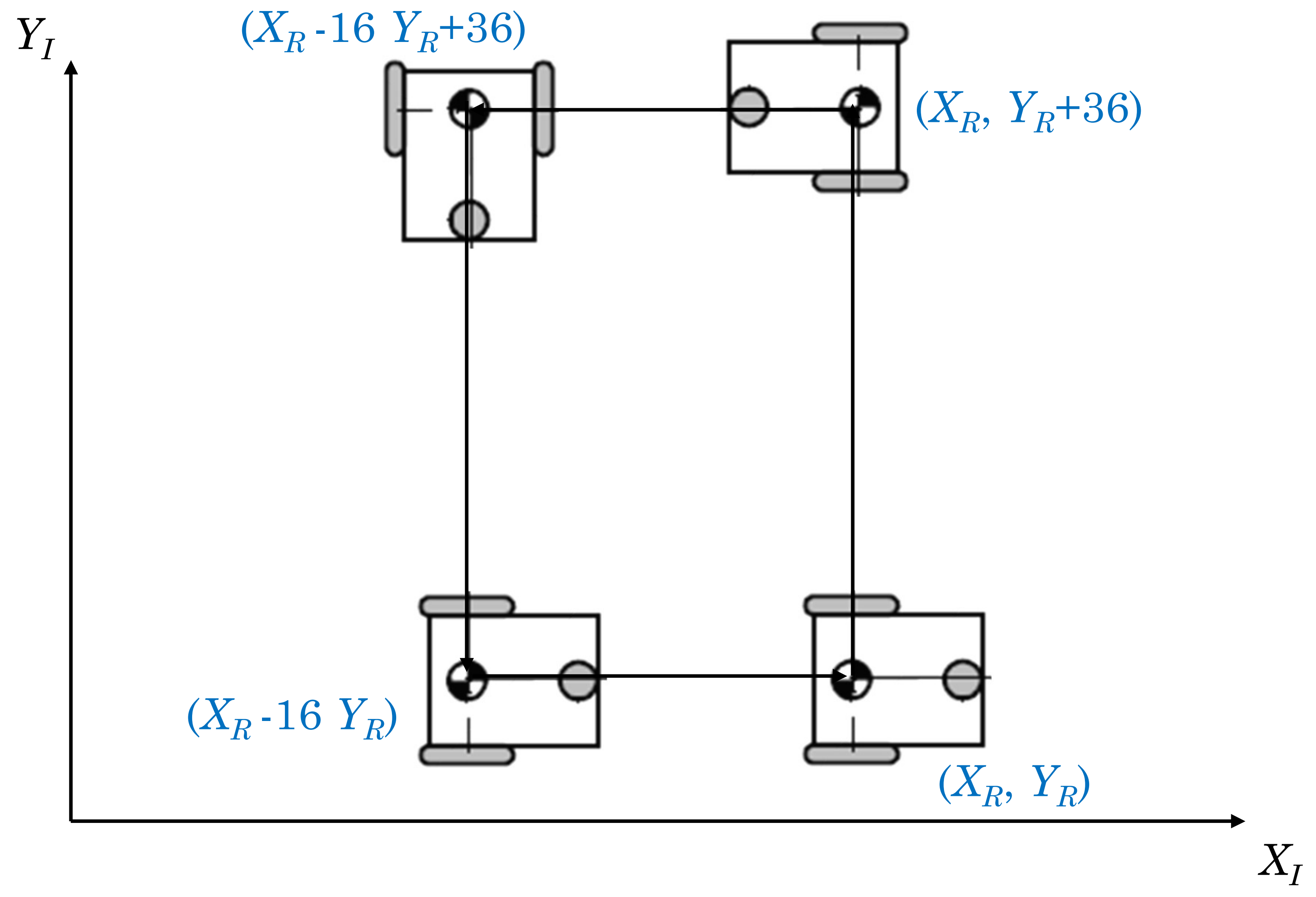
1. Initial location
   - (X_R, Y_R)
2. Forward 36 in
   - (X_R, Y_R+36)
3. Pivot left 90°
   - (X_R, Y_R+36)
4. Forward 16 in
   - (X_R-16, Y_R+36)
5. Pivot left 90°
   - (X_R-16, Y_R+36)
6. Forward 36 in
   - $(X_R-16\ Y_R)$

Kinematics & Localization



1. Initial location
   - $(X_R,\ Y_R)$
2. Forward 36 in
   - $(X_R,\ Y_R+36)$
3. Pivot left 90^o
   - $(X_R,\ Y_R+36)$
4. Forward 16 in
   - $(X_R-16,\ Y_R+36)$
5. Pivot left 90^o
   - $(X_R-16,\ Y_R+36)$
6. Forward 36 in
   - $(X_R-16\ Y_R)$
7. Pivot left 90^o
   - $(X_R-16,\ Y_R)$

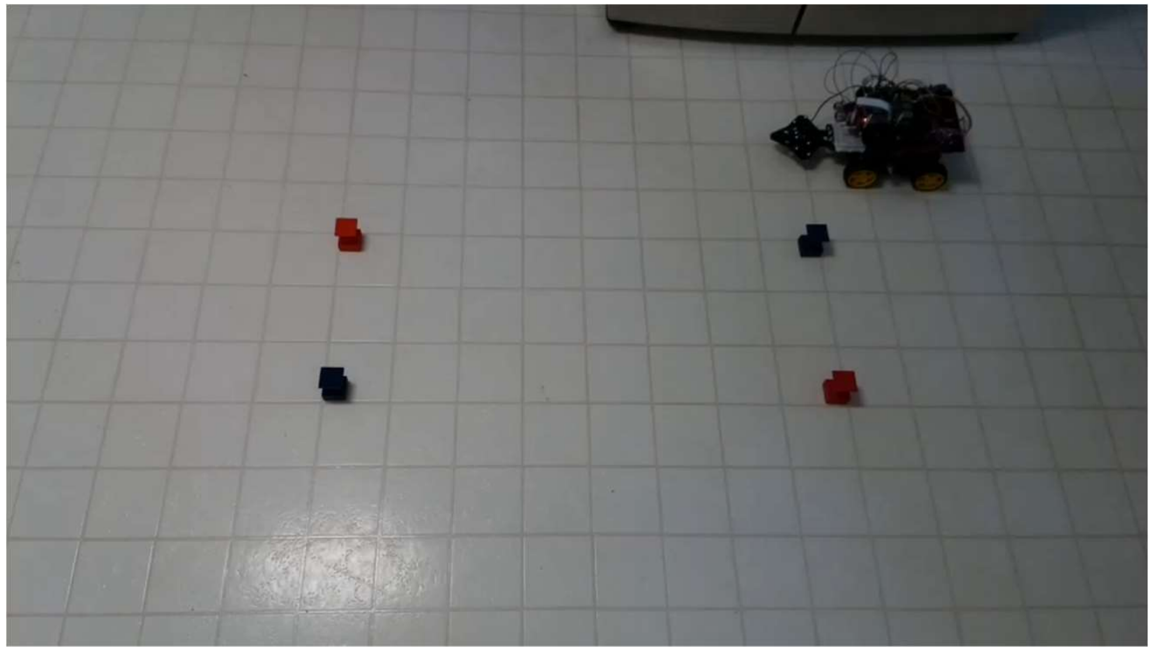
Kinematics & Localization

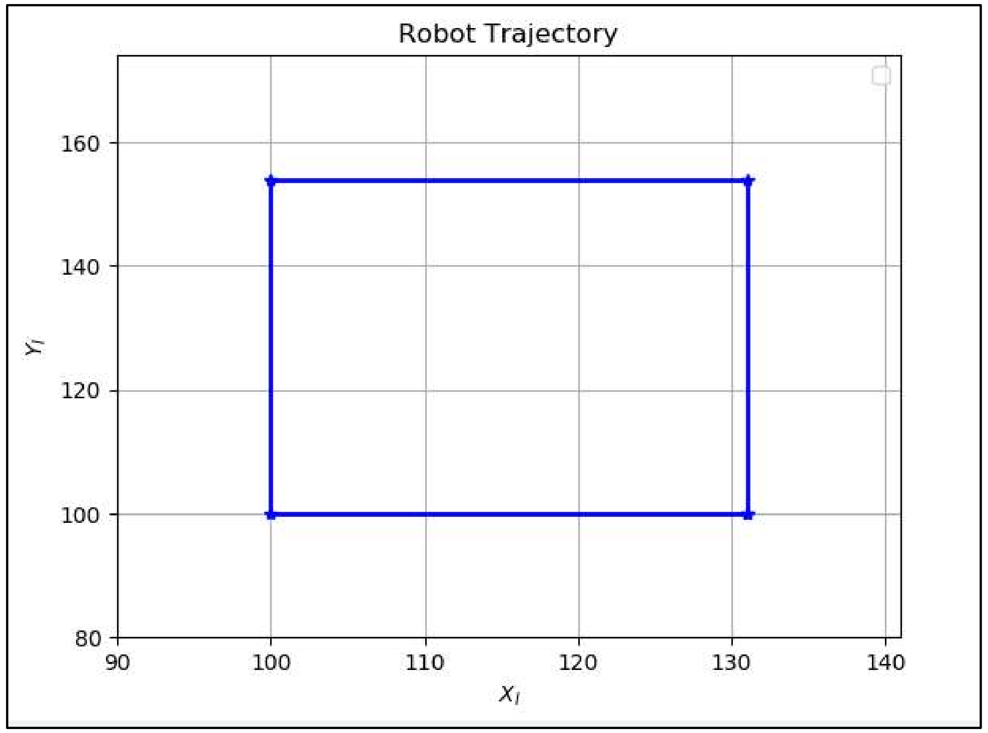


1. Initial location
   - (X_R, Y_R)
2. Forward 36 in
   - (X_R, Y_R+36)
3. Pivot left 90°
   - (X_R, Y_R+36)
4. Forward 16 in
   - (X_R-16, Y_R+36)
5. Pivot left 90°
   - (X_R-16, Y_R+36)
6. Forward 36 in
   - $(X_R-16\ Y_R)$
7. Pivot left 90°
   - (X_R-16, Y_R)
8. Forward 16 in
   - (X_R, Y_R)

In-Class Exercise

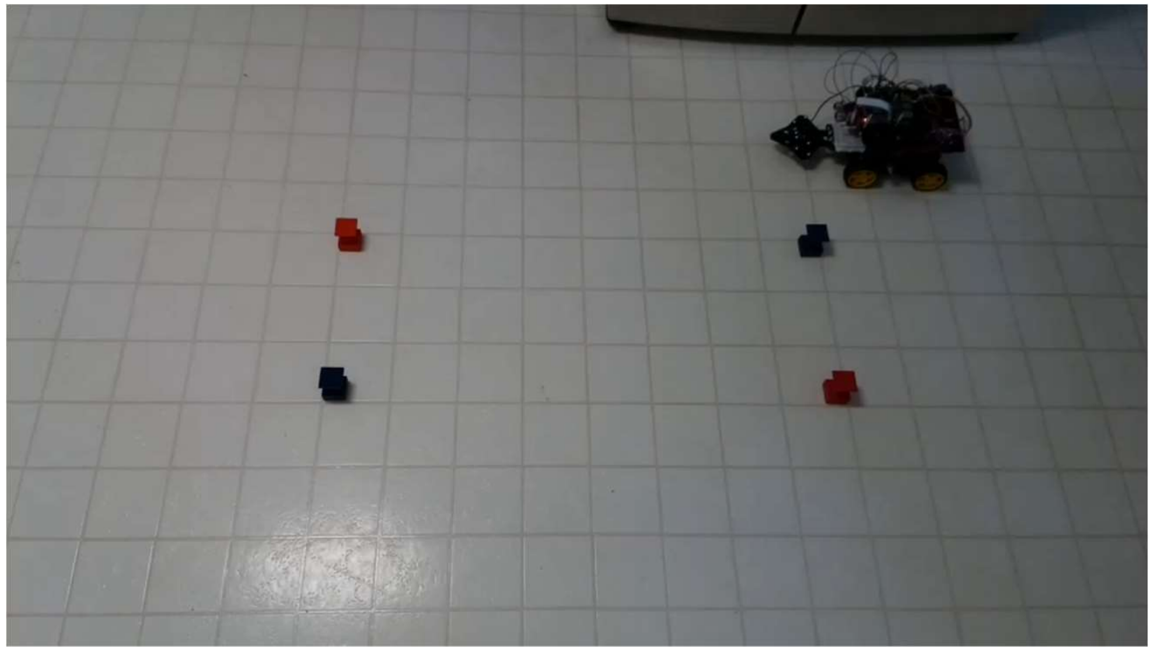
- Create new Python script ***map01.py***
- Script must:
  1. Drive robot in <u>rectangular</u> pattern
  2. Record position data
- Once complete, open & plot position data in Matplotlib

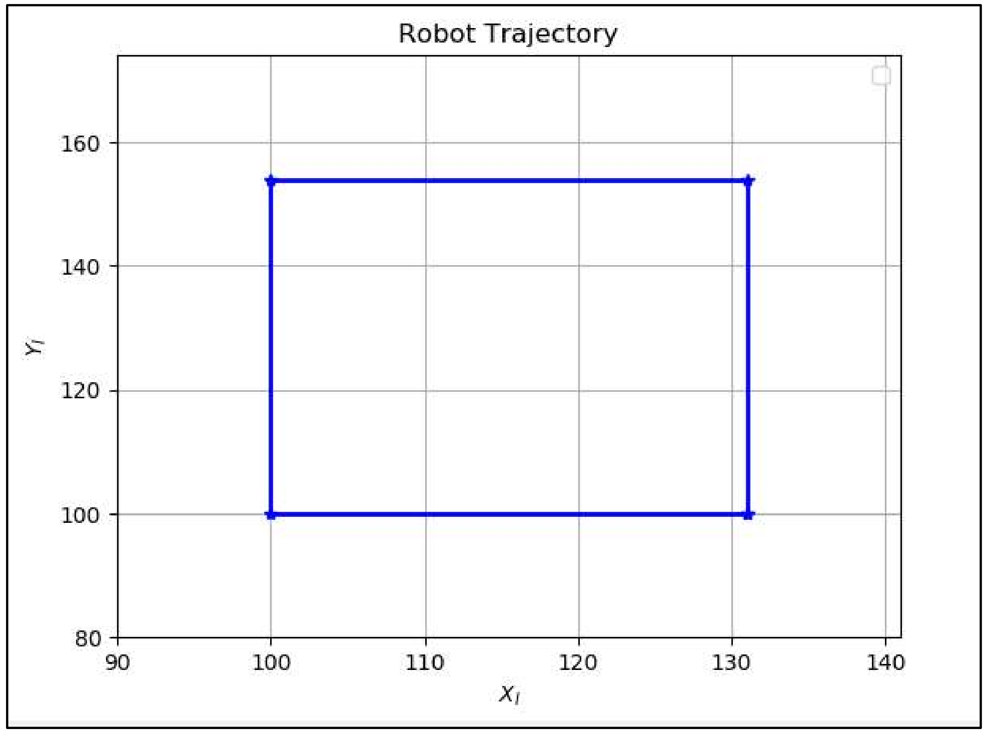




In-Class Exercise

- Create new Python script ***map02.py***
- Script must:
  1. Take as input a sequence of commands from user
  2. Drive robot through sequence
  3. Record position data through sequence
- Once complete, open & plot position data in Matplotlib





References

- *Introduction to Autonomous Mobile Robots*, Siegwart
  - Chapter 5
- *SSMTP*
  - https://wiki.archlinux.org/index.php/SSMTP
- Send *email from a Python script on the Raspberry Pi*, Gaven MacDonald
  - https://www.youtube.com/watch?time_continue=13&v=0kpGcMjpDcw