DEFENDER
Modular bicycle security system
ECE 110/120 Honors Lab
Junyang Zhou  junyang5
Hongxuan Li    hl3

# Introduction

## Statement of Purpose

Owning a fantastic bike is a cool thing, but when someday you walk out your classroom and find out that the wheel or seat of your bike is missing, that is not cool at all. The construction design of modern bicycle caused it very easy to be disassembled. Adding extra lock could reduce the chance of bicycle parts, especially front wheels and seats, being stolen. However, using multiple locks also increases time taken to unlock the bike, which is inconvenient.

Now, the Defender is a special bicycle lock which was designed to help people with such dilemma. With this lock, we hope people could park their bike safely in any place without worrying about its wheels or seat being stolen, or bringing multiple locks around. We designed different locks for different parts of the bike and used bluetooth to connect them together.  As we further explore and develop our design and prototype, we decided to add alarm system as extra safety insurance, and we plan to add a tracking system and its mobile app suite so user can be notified if any person is trying to break the lock, and be able to track their lost bike.

## Features and Benefits

- Automatic unlocking of frame locks when main lock is unlocked
- Alarm when lock is being hacked for more than 3 seconds
- Extra security for the front and back wheel
- Strong U-lock

# Design

## System Overview

### Main Lock

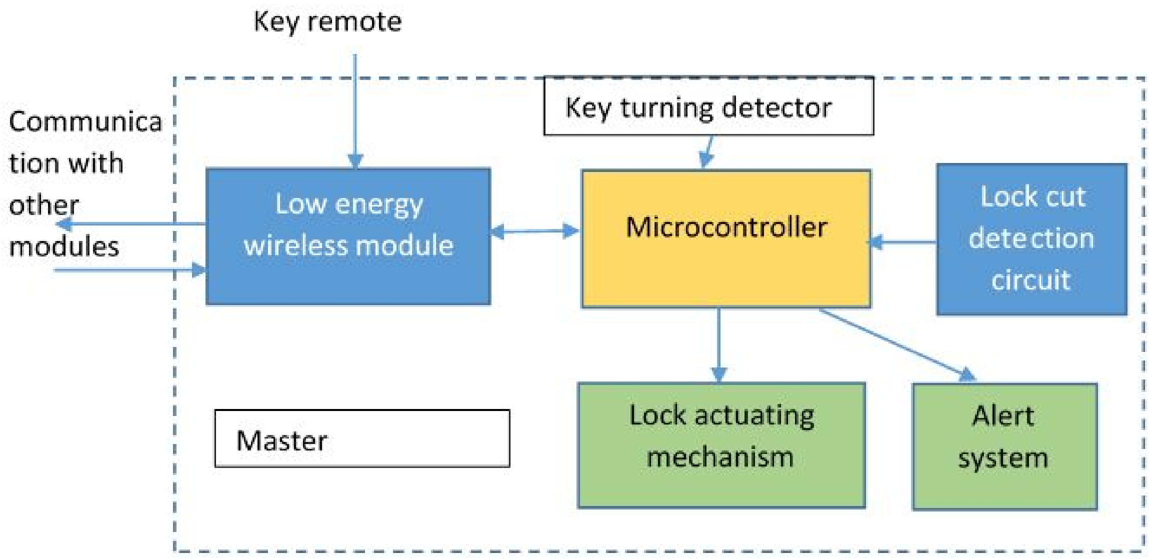
Have the alarm system and keyhole.
Sends signal to all paired locks to sync with the status of main lock.

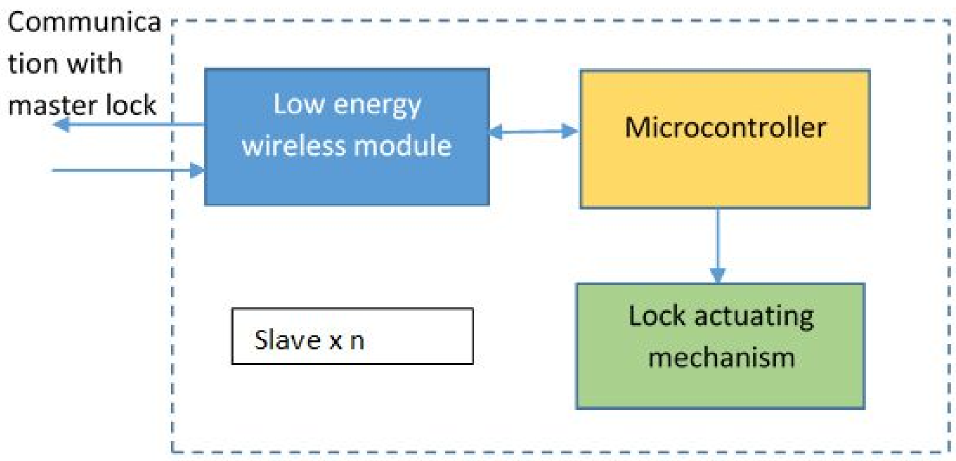
### Frame lock/secondary locks

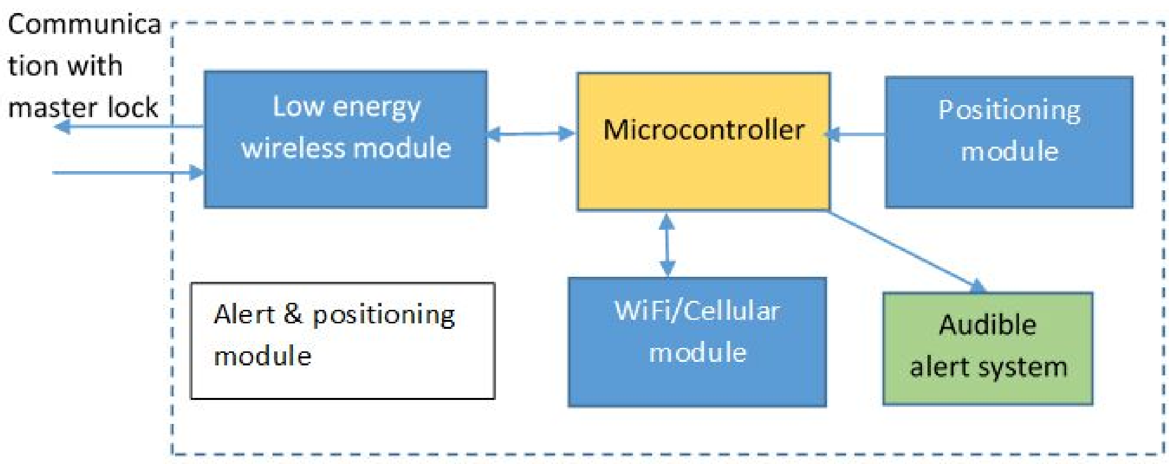
Sync with the status of main lock.
If main lock is in unlocked state, turn servo to unlock.
If main lock is in locked state/alarm state, if hall effect sensor senses magnet, turn servo to latch.

### Alert and positioning system

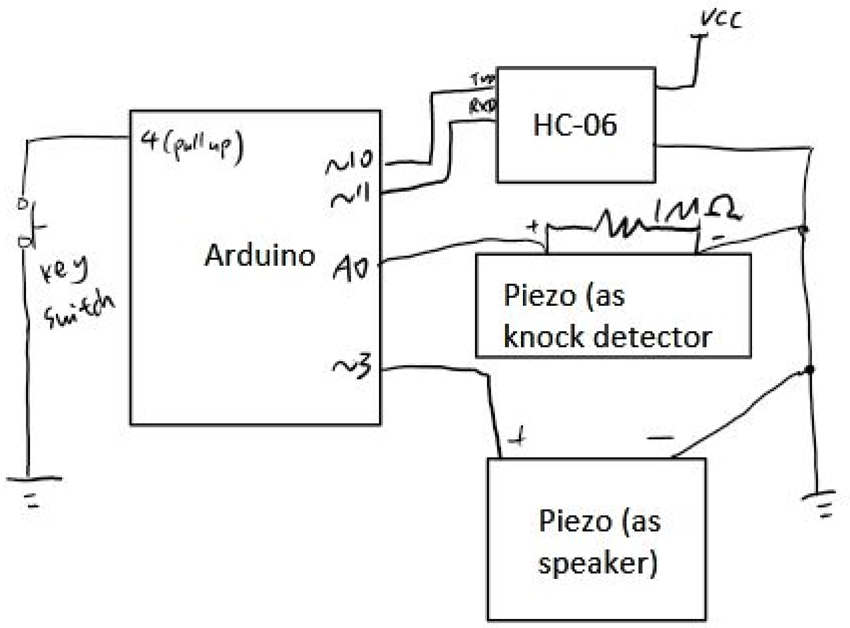
Sends alert on user's phone if main lock goes to alarm state.
Begin tracking bike location when main lock activates alarm.

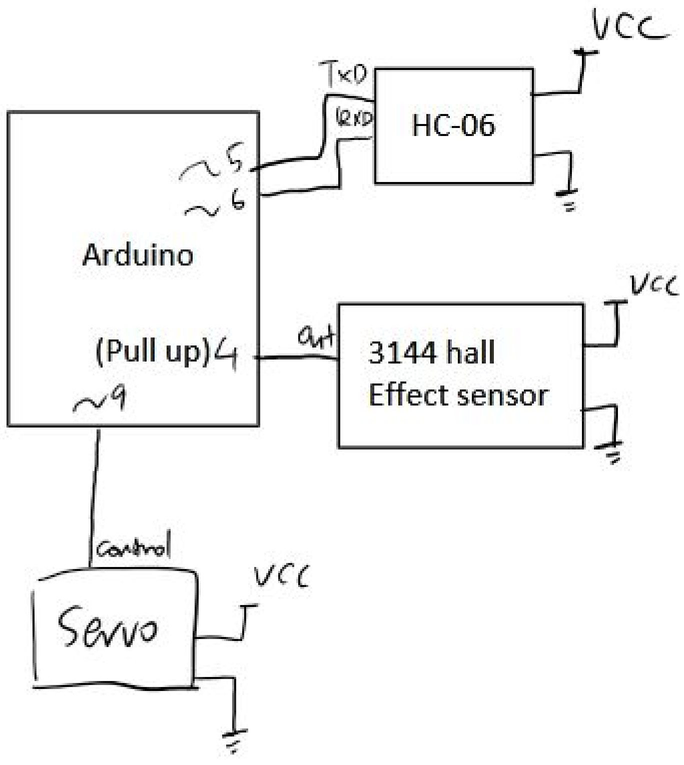

Key remote
Communication with other modules
Key turning detector
Low energy wireless module
Microcontroller
Lock cut detection circuit
Master
Lock actuating mechanism
Alert system


Communication with master lock
Low energy wireless module
Microcontroller
Slave x n
Lock actuating mechanism

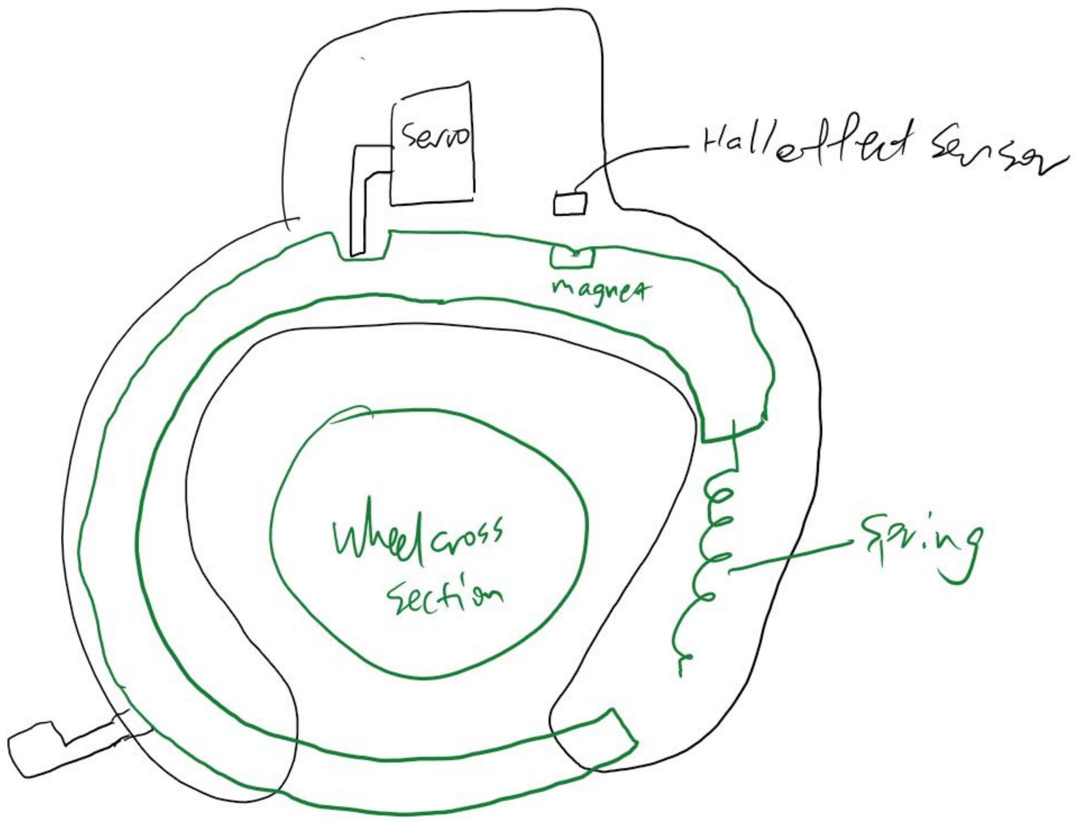

Communication with master lock
Low energy wireless module
Microcontroller
Positioning module
Alert & positioning module
WiFi/Cellular module
Audible alert system

## Design Details

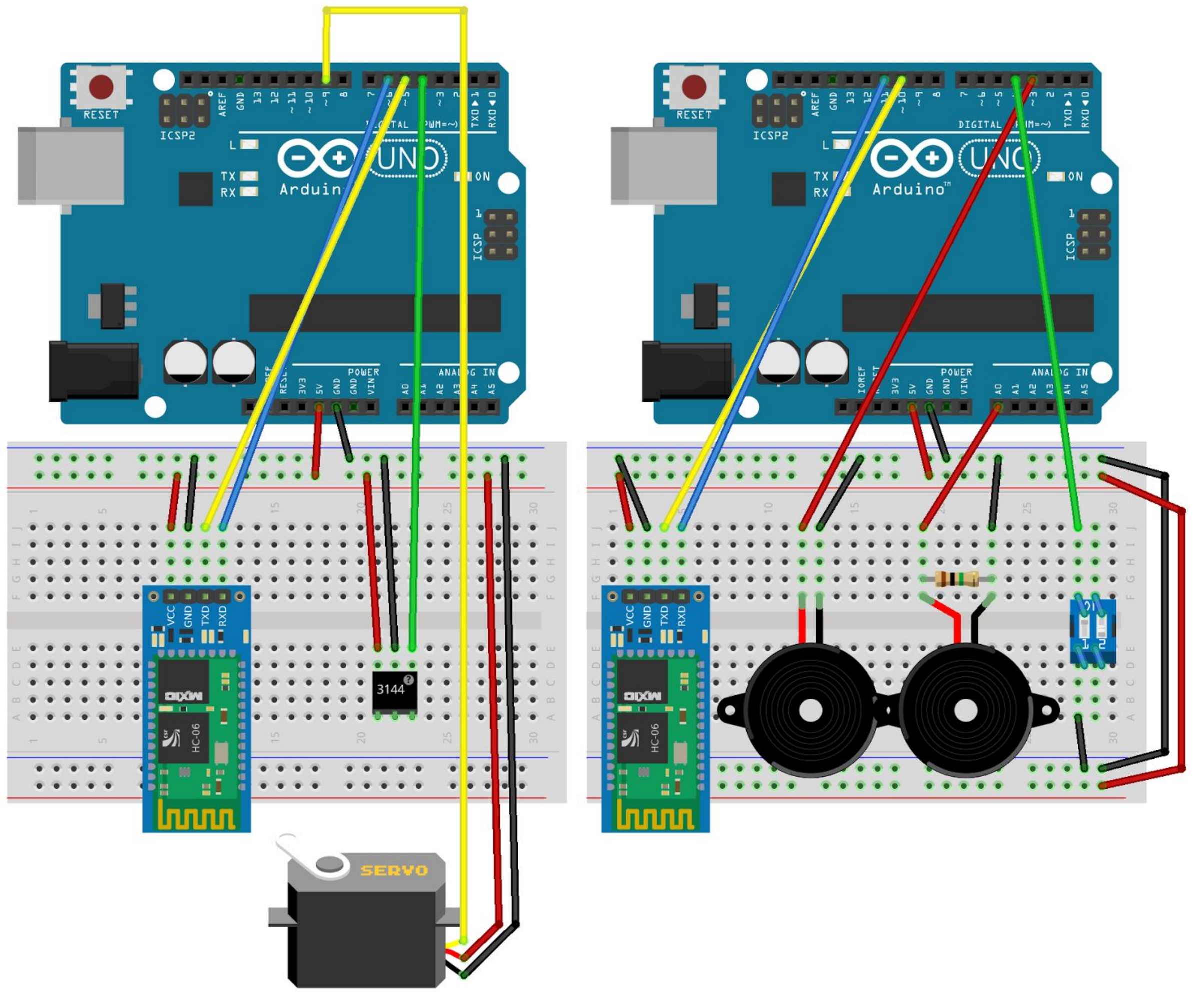




Frame lock internals



## Wiring



HC06 board from https://github.com/RafaGS/Fritzing/blob/master/Bluetooth%20HC-06.fzpz

fritzing

## Results

Piezo responds to very light taps. We set the threshold of AnalogRead value to 10 in the end.
Hall effect sensor responds to magnet well.
Servo is very shaky, probably due to insufficient voltage supply or timer conflict.

## Problems and Challenges

a. The first challenge that confronts us is the building of the bluetooth connection between two bluetooth module. After we finished the coding of the bluetooth connection, the module doesn't perform the task as we thought, instead of building a double-side connection, the module appears that only the master module could receive information, not the slave module. After carefully examined our code, we can guarantee that it's undoubtedly correct. This left us no choice but to reset them. However, after being reseted, the module start to perform the normal double-side connection. The reason for this is still unknown.
b. The second challenge of us is to get the servo work normally, since at first our servo seems to be only able to turn in one direction. At first we thought such situation is due to some shortcut happened inside the servo but it turns out there is something wrong with MacBook Air's USB port. The servo works if Arduino board is powered by a Windows laptop somehow.
c. The third challenge is to build the alarm system of the lock. The point is to set the sensitivity of the piezo to a certain level which could prevent making noises and perform the task of alarm as well. It is also a challenge to set an exact timeframe for detection.

## Future Plans

Develop the alert and tracking system with its mobile app.

## References

Bluetooth tutorial
http://www.instructables.com/id/Modify-The-HC-05-Bluetooth-Module-Defaults-Using-A/?ALLSTEPS

HC-06 Fritzing part
https://github.com/RafaGS/Fritzing/blob/master/Bluetooth%20HC-06.fzpz

AXA Defender frame lock
http://www.axasecurity.com/bike-security/en-gb/products/locks/framelocks/axa-defender-silver/

Bike lock security comparison http://thebestbikelock.com

Piezo sensor tutorial https://www.arduino.cc/en/Tutorial/Knock

3144 Hall effect sensor datasheet
http://www.elecrow.com/download/A3141-2-3-4-Datasheet.pdf

Servo library reference https://www.arduino.cc/en/Reference/Servo

# Appendix

## Arduino code for frame lock

```
#include <Servo.h>
#include <SoftwareSerial.h>

Servo myservo; // create servo object to control a servo
SoftwareSerial BTSerial(5, 6); // RX | TX
int hallepin = 0; // hall effect sensor
int val; // lock state

void setup() {
  myservo.attach(9);
  pinMode(4,INPUT_PULLUP);
  Serial.begin(38400);
  BTSerial.begin(38400);
}

void loop() {
  Serial.print(digitalRead(4));
  hallepin=digitalRead(4);
  val = BTSerial.read();
  Serial.println(val);
  if(val==0) {//if main lock unlocked
    myservo.write(0);//turn servo to unlock angle
  }

  else if(hallepin==0) {//if main lock is in all other state
(lock,
                        //alarm, disconnected) and hall effect
                        //sensor senses magnet
    myservo.write(90); //turn servo to lock
  }
  delay(15);
}
```

Arduino code for the master lock

```
#define keyPin 4 //switch used as key
#define piezoPin A0 //piezo detector input
#define speakerPin 3 //piezo output
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10, 11); // RX | TX
int piezo=0;
int lock=1;//0-unlock,1-lock,2,alarm

unsigned long alarmStartTime=0;
unsigned long runtime=0;
int freq=1000;//frequency of alarm sound played
bool freqUp=1;
bool alarmStarted=0;
void playAlarm(){
  if(lock==2&&!alarmStarted){
    alarmStartTime==millis();
    alarmStarted=1;
  }
  if(lock==0){
    noTone(speakerPin);
    alarmStarted=0;
  }
  if(alarmStarted){
    runtime=millis()-alarmStartTime;
    if(runtime>=30000){
      noTone(speakerPin);
      alarmStarted=0;
      lock=1;
      return;
    }
    else{
      tone(speakerPin,freq);
      if(freqUp)freq+=50;
      else freq-=50;
      if(freq>=3000)freqUp=0;
      else if(freq<=1000)freqUp=1;
    }
  }
}

int secCounter=0;
int piezoDetCounter=0;
int piezoDetected = 0;
```

```
void piezoDetection(){
  piezo=analogRead(piezoPin);
  if(piezo>10)piezoDetected=1;
  if(millis()%1000<10){
    secCounter++;
    piezoDetCounter+=piezoDetected;
    piezoDetected=0;
  }
  if(secCounter>3){
    if(piezoDetCounter>2&&lock==1)lock=2;
    secCounter=0;
    piezoDetCounter=0;
  }
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(38400);
  BTSerial.begin(38400);  // HC-05 default speed in AT command
more 38400
  pinMode(speakerPin,OUTPUT);
  pinMode(keyPin,INPUT_PULLUP);//0-unlocked, 1-locked
}

int keyState=1;
void loop() {
  // put your main code here, to run repeatedly:
  keyState=digitalRead(keyPin);
  if(lock!=2){
    lock=keyState;
  }
  else if (keyState==0){
    lock=keyState;
  }
  piezoDetection();
  playAlarm();
  if(piezo>20)Serial.println(piezo);
  Serial.println(lock);//for debugging
  BTSerial.write(lock);
}
```