Autonomous Delivery Robot

Devang Dave[1], Parth Parsana [2], Aarjav Ajmera[3]

[1,2,3]B.Tech. Student, Department of Electronics and Telecommunication Engineering, SVKM's NMIMS MPSTME, India

---

Abstract - *The autonomous delivery robot is meant to be a substitute for a goods delivery person. The delivery robot is capable of navigating through a cluttered space environment from a home location to a destination point while avoiding obstacles in the process. It uses an ultrasonic sensor to detect if anything has been placed inside the bin and only once when something is placed inside, the robot starts moving from a position to another.*

Bluetooth beacons are used to represent the start, end and mid-point which will be used by the robot to differentiate the specified locations. Bluetooth beacons will act as a terminal for the robot to detect. The autonomous robot will move on an undesignated path i.e. an unmarked route unlike the archaic robots only capable of moving inside a marked black lane. The robot will be trained and multiple simulations are run on it with the help of the DonkeyCar[1] library so it can successfully avoid obstacles and relay the path with efficacy, making the delivery fast and efficient. A Pi camera is used which will help rectify the unmarked desired path over which the robot has to move.

1. INTRODUCTION

In this present era, with businesses booming we see a rapid growth in companies whether it be a comparison financially or in ground strength. For a company to work efficiently it has been repeatedly observed that multiple personnel are used for piteous jobs such as transferring documents to various departments or delivery of letters or posts. So, proposing an autonomous robot capable to send or receive physical items, in a number of different scenarios.

Autonomous delivery system, not yet intelligent enough to deliver goods across cities, but able to deliver small objects from one place to another in small boundaries. The use of DonkeyCar[1] represents the robot, open source platform, which will be powered by the Raspberry Pi and Pi Camera to create an autonomous vehicle. The delivery robot is trained meticulously with support of OpenCV[2] and TensorFlow[3], which are a part of DonkeyCar[1] library to allow run simulations and can be used to teach or create autopilots or models. Thus, allowing the car to manipulate itself by the judgment bestowed upon it through rigorous training.

1.1 Motivation and Scope

On a recent visit to a leading financial hub, a particular aspect that struck us the most during the visit was that although the firm was pretty new and quite technically advanced, paper was invariantly the most opted form for communication. Since the firm has a huge capacity with quite a lot of employees, the office boy seemed a common show for transporting the important paper/material around. While they were inexplicably good at their job it seemed invariant to call them every time for small scale jobs such as the delivery of notes, packages, paper material from one department to another.

Although the use of autonomous robots is not a new advance in the field of delivery as multinationals companies such as Amazon and DHL have been known to use such robots for the purpose to handling packages in their warehouses, it seemed quite favorable to propose something of such equivalence particularly in the eye of such firms and companies.

Autonomy can be scaled to various prospects depending on the usage. In our project, our initial step is to introduce a robot capable of carrying out minimalistic jobs mentioned earlier. Teaching, specifically training the robot to travel is the most challenging section of the project. Our main focus was to teach the robot about obstacle avoidance as well as navigating to the destination with accuracy. With a heavy data set on us, we can implement obstacle avoidance as well as reach our final desired destination with the marginal error.

1.2 Problem statement

The goal is to overcome the problem of manually delivering good from a place to place. It saves time and makes a workplace more efficient. The objective is to setup an autonomous robot to stop at certain points and for the robot to understand when mobility is needed. So, an ultrasonic sensor is connected to the delivery robot to make the car respond only when it detects certain material in the bin.

2. METHODOLOGY

How the Autonomous Delivery Robot works: -

At first, the Delivery robot should be waiting at the origin. It should detect (using the ultrasonic sensor) when some material is placed in the container (on the Delivery robot). Once detected, it should start moving on a specified track. Once the user notices the package and picks it up from the container on the robot, the sensor should detect that there is nothing to deliver at the moment. Therefore, it should reach the origin and stop there, waiting for another letter, and the cycle continues.

How the letter detection works:

The HC-SR04 ultrasonic sensor is connected to the Arduino MKR1000 which has been placed on the Delivery robot. It gives the distance in centimeters. The ultrasonic sensor is placed in the container in such a way that there is a noticeable difference in the distance it gives when the container is empty and when a letter is placed.

How the Bluetooth beacon works:

The HC-05 Bluetooth module which has been connected to the MKR1000 on the delivery robot will indicate with the help of state pins [4]. It will go HIGH when a Bluetooth connection with another Bluetooth device which we will be calling the Bluetooth beacon is established and LOW when it is not connected. We will use digitalRead() [5] with the MKR1000 on this pin to detect if the delivery robot is at the origin or not. As for the beacon, it could theoretically be any Bluetooth device. So, to summarize, the HC-05 is wired to the MKR1000 and configured to automatically connect to the Bluetooth Beacon if it is within range, and the MKR1000 can detect whether or not the HC-05 is connected to anything (using the state pin as an indicator). This information will be sent to the Raspberry Pi, which can then finally detect whether or not it is at the origin.

2.1 Hardware

Raspberry Pi 3b+ (with SD card): - The delivery robot uses the Raspberry Pi as the brain for the project, it runs the DonkeyCar[1] library. Firstly, we will need install the software and DonkeyCar libraries on the Raspberry Pi, as well as on another host PC. The host PC is needed because the Raspberry Pi is not powerful enough to train an autopilot. The training is done using Kera's TensorFlow which comes incorporated within the DonkeyCar library. Upon doing so, we get an autopilot file which we will need to transfer back to the Raspberry Pi. We can use this autopilot file to drive the delivery robot within the planned route needless of marking the area and increasing the accuracy as we strengthen our dataset.

Fig 2.1.1 Raspberry Pi 3b+

Wide Angle Raspberry Pi Camera: -

Raspberry Pi camera module v2 as shown in Figure 2.1.2 is used to record the images to capture the dataset as well as to keep a visual track of the surrounding while the car is driving autonomously.

Fig 2.1.2 Wide Angle Raspberry Pi Camera

Arduino MKR1000: -

The Arduino MKR1000 used here is used for determining whether the robot is in the required place. It a WIFI development board which is interfaced with the Bluetooth module and ultrasonic sensor and placed on our delivery robot.

Fig 2.1.3 Arduino MKR1000 ARMX0004

Ultrasonic Sensor Module: -

This module is used to detect the distance between two surfaces or it can even be used to detect objects placed in the bin. Figure 2.1.4 and 2.1.5 show the HC-SR04 module and its pin layout.

Fig 2.1.4 Ultrasonic module (HC-SR04)

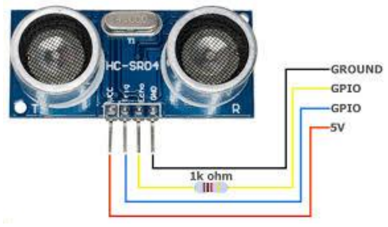


Fig 2.1.5 HC-SR04 Pinout diagram

Bluetooth RF Transceiver Module: -

Use of HC-05 module allows us to connect with the MKR1000 to identify the certain destination points in an area. Figure 2.1.6 below represents the Bluetooth Module.

Fig 2.1.6 Bluetooth HC-05 Transceiver Module

Steering and Throttle Control: -

There are two motors in our delivery robot to control its speed and direction: a DC brushed motor to control the speed of all the 4 wheels and a servo motor to control the steering. The motors are controlled using PWM/servo driver PCA9685 as shown in Figure 2.1.5 It uses I2C communication [6]. The servo motor is directly controlled using the PCA9685 servo driver while the throttle output of servo driver is first given to an electronic speed controller (ESC) which then controls the speed of the throttle motor, as shown in Figure 2.1.4.

Fig 2.1.4 Electronic Speed Controller (ESC)

PWM/servo driver PCA9685: -

There will be a need to calibrate [7] our Delivery Robot steering and throttle so as to make it work consistently. To do that, find the optimal PWM values for the throttle and steering and use the PCA9685 to feed our calibrated values to the ESC which then passes it forward the motor for throttle.

Fig 2.1.5 Servo Driver PCA9685

Joystick Controller: -

To train the RC car run autonomously, we require the dataset to train the model. That dataset is captured first by driving the car manually and recording the data of all the sensors which can later be used to train the model. Joystick controller is required to run the car manually. Sony PlayStation Dual Stick wireless gamepad is used which is shown in Figure 2.1.6.

Fig 2.1.6 PS4 Wireless Dual Shock Controller

HSP 94168 (RC Car): -

It is a 1/10th scale car. The main reason to use this RC car is because it has separate ESC, Receiver, Transmitter and brushed RC motor.

Fig 2.1.7 HSP 94186 RC car

Fig 2.1.8 HSP 94186 RC car

3D Printed Surface Mount: -

To safely place our components on the RC car of our delivery robot, a 3D model is built whose soft copy was available on the DonkeyCar community. Figure 2.1.9 shows the representations of our 3D mount. The bottom chassis comes with prepositioned perforations which help with ease to attach the several components.

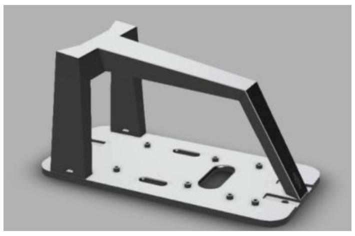


Fig. 2.1.9 3D printed chassis

2.2 Implementation

Model Training

To produce a predictive model [8], we train our database with the help of the Keras high level api [9]. They are intended to be used with the TensorFlow backend. Keras consists of multiple parts which are designed to use the trained artificial neural network to reproduce the steering and throttle values with the help of live feed from the mounted camera on the delivery robot.

For our robot, we had chosen the Keras Categorical learning network [9]. The neural network pilot/model breaks the steering and throttle decisions into discreet angles and then uses categorical cross entropy to train the network to activate a single neuron for each steering and throttle choice. It transforms continuous real numbers into a range of discreet values for training and runtime.

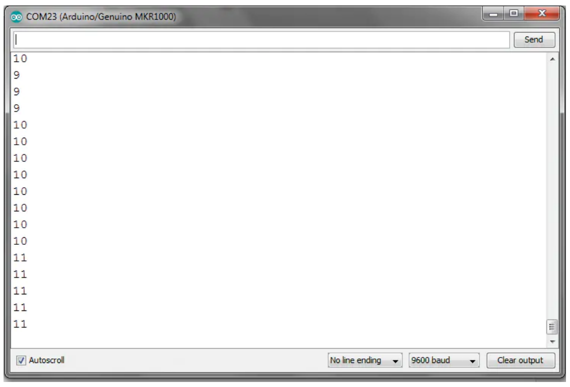
Configuring the Bluetooth module and the Bluetooth Beacons:

We Configure our Bluetooth module, connected to our delivery robot which is responsible for connecting with Bluetooth beacons by putting the HC-05 Bluetooth Module in AT mode [10] and uploading of the "AT Command" sketch [10] to the Arduino MKR1000. With the help of this process, it allows us to use the Arduino MKR1000 to give commands to the Bluetooth module in AT mode.

Configuring the ultrasonic sensor and the setting up the detection

The ultrasonic sensor is placed in the container in such a way that there is a noticeable difference in the output when the container is empty, and when a letter is placed in it. To accomplish this, we implemented a code to find the output of the ultrasonic sensor in centimeters.

Figure 2.2.1 shows the output values on the serial com. Figure 2.2.2 shows the Bluetooth module and the ultrasonic sonic sensor mounted on a container which is placed on our delivery robot and will be used for placing letters.



```
10
9
9
9
10
10
10
10
10
10
10
10
11
11
11
11
11
```

Fig. 2.2.1 Serial Com output values

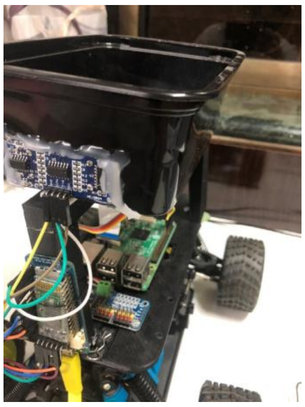


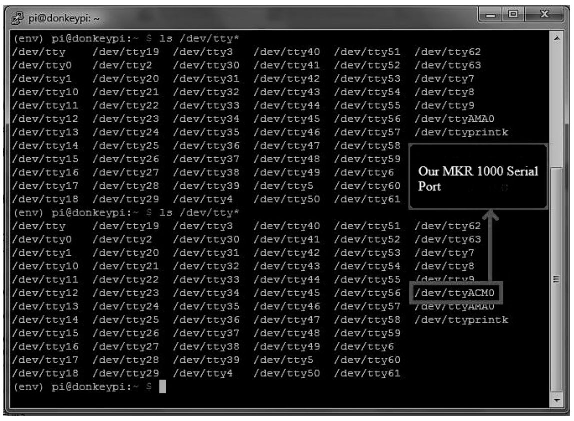
Fig. 2.2.2 Bluetooth beacon and ultrasonic sensor mounted on the delivery robot

Modifying the Robot:

We created a new part to work with the DonkeyCar library, which will allow the Raspberry Pi to retrieve data from the MKR1000 and decide the autonomous delivery system's behavior based on that data.

First, we need to know the serial port the MKR1000 will connect to on the Raspberry Pi. So, to do so we open an

SSH terminal to connect to our Raspberry Pi and look for the serial port MKR1000 in connected to. Figure 2.2.3 shows the serial port MKR1000 is connected to our Rpi.


pi@donkeypi: ~

```
(env) pi@donkeypi:~ $ ls /dev/tty*
/dev/tty    /dev/tty19  /dev/tty3   /dev/tty40  /dev/tty51  /dev/tty62
/dev/tty0   /dev/tty2   /dev/tty30  /dev/tty41  /dev/tty52  /dev/tty63
/dev/tty1   /dev/tty20  /dev/tty31  /dev/tty42  /dev/tty53  /dev/tty7
/dev/tty10  /dev/tty21  /dev/tty32  /dev/tty43  /dev/tty54  /dev/tty8
/dev/tty11  /dev/tty22  /dev/tty33  /dev/tty44  /dev/tty55  /dev/tty9
/dev/tty12  /dev/tty23  /dev/tty34  /dev/tty45  /dev/tty56  /dev/ttyAMA0
/dev/tty13  /dev/tty24  /dev/tty35  /dev/tty46  /dev/tty57  /dev/ttyprintk
/dev/tty14  /dev/tty25  /dev/tty36  /dev/tty47  /dev/tty58
/dev/tty15  /dev/tty26  /dev/tty37  /dev/tty48  /dev/tty59
/dev/tty16  /dev/tty27  /dev/tty38  /dev/tty49  /dev/tty6
/dev/tty17  /dev/tty28  /dev/tty39  /dev/tty5   /dev/tty60
/dev/tty18  /dev/tty29  /dev/tty4   /dev/tty50  /dev/tty61
(env) pi@donkeypi:~ $ ls /dev/tty*
/dev/tty    /dev/tty19  /dev/tty3   /dev/tty40  /dev/tty51  /dev/tty62
/dev/tty0   /dev/tty2   /dev/tty30  /dev/tty41  /dev/tty52  /dev/tty63
/dev/tty1   /dev/tty20  /dev/tty31  /dev/tty42  /dev/tty53  /dev/tty7
/dev/tty10  /dev/tty21  /dev/tty32  /dev/tty43  /dev/tty54  /dev/tty8
/dev/tty11  /dev/tty22  /dev/tty33  /dev/tty44  /dev/tty55  /dev/tty9
/dev/tty12  /dev/tty23  /dev/tty34  /dev/tty45  /dev/tty56  /dev/ttyACM0
/dev/tty13  /dev/tty24  /dev/tty35  /dev/tty46  /dev/tty57  /dev/ttyAMA0
/dev/tty14  /dev/tty25  /dev/tty36  /dev/tty47  /dev/tty58  /dev/ttyprintk
/dev/tty15  /dev/tty26  /dev/tty37  /dev/tty48  /dev/tty59
/dev/tty16  /dev/tty27  /dev/tty38  /dev/tty49  /dev/tty6
/dev/tty17  /dev/tty28  /dev/tty39  /dev/tty5   /dev/tty60
/dev/tty18  /dev/tty29  /dev/tty4   /dev/tty50  /dev/tty61
(env) pi@donkeypi:~ $
```

Figure 2.2.3 SSH terminal output when new serial port connected

We create a python file in our local directory in the Raspberry pi. We want this code to run whenever we run our model. Rpi will send a bit "1" at recursive period to MKR1000. In response, the MKR1000 will give the Rpi two values in terms of "1" and "0", which will define whether a letter is detected or not and whether a Bluetooth Connection is made between the Bluetooth Module and a Bluetooth Beacon.

To do so, we tweak our previous codes which gave us the values of the ultrasonic sensor and a connection establishment between Bluetooth Module and a Bluetooth Beacon to a single code which now gives us a bit value output for the same. The figure below shows the reference. Therefore, this program provides us with an output which tells us about the status of the robot while on the move

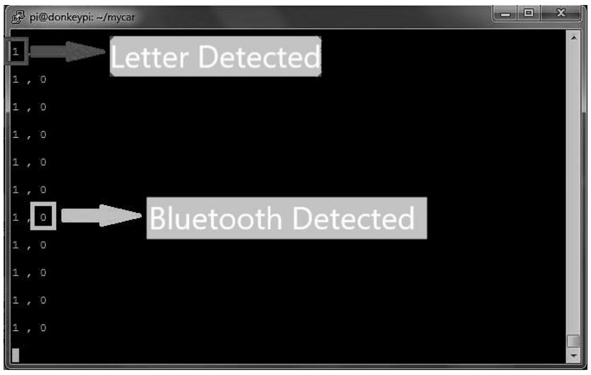
3. Results

Testing the Bluetooth Module and Bluetooth Beacon connection:

We tested this step and its success by turning on a Bluetooth Beacon placed at a point and bringing the Bluetooth beacon to its proximity step by step. The red LED on both the Bluetooth Module and Bluetooth Beacon will blink very slowly and in sync with each other once every couple of seconds if a successful recognition and pairing of the Module and Beacon takes place. During our testing phase, we were successfully able to achieve this allowing us to move onto the next step.

Testing of the updated part added to the delivery robot:

With the help of the updated part, the Rpi can receive feedback on whether a letter has been placed inside the bin or not, also whether the robot has reached the desired destination. With the help of the code written on the MKR1000, the Rpi can determine the status of the car during a run by showing feedback on the terminal with the help of two bits as shown in figure 3.1 below. When a letter is placed in the bin, the first bit changes from '0' to '1' whereas the same happens with the second bit when a Module to Beacon connection is made upon reaching the destination. These values are continuously refreshed after a moment of elapsed time.


pi@donkeypi: ~/mycar

```
1
1 , 0
1 , 0
1 , 0
1 , 0
1 , 0
1 , 0
1 , 0
1 , 0
1 , 0
1 , 0
```

Figure 3.1 Letter and Bluetooth detection

4. Advantages, Limitations and Applications

Advantages

Using our project, we can substitute the job of a delivery person since this robot is capable of delivering goods from one place to another which can thus, help save money over time as this option will be cheaper for a company to use for delivery purposes in the long run. It can also help reduce man-power which in turn will help save energy, time and the problem of availability of an individual in times of need. The robot has onboard sensors and cameras to operate and which makes the wires or magnetic tape obsolete. Instead of following hard-set paths, the robot can dynamically create their efficient pathways from Point A to Point B within a facility, helping them to avoid obstacles, hence increasing the flexibility. Since it uses Machine Learning, the delivery robot improvises every time it takes a similar or different route which increases its repeatability and accuracy bringing quickness to implementation.

Limitations

Firstly, the robot can't be used for delivery over long distances or over the roads keeping precarious roads caused due to holes might create problems for the mobility of the car. Secondly, the making of the robot is a lengthy procedure since a lot of training with regards to not only the precision of the model but also the obstacle avoidance feature is required urging an eclectic and abundant data and collection of which can be a very time-consuming procedure. Thirdly, the current project agenda doesn't comprise of a locking mechanism leaving the material susceptible to theft during the of delivery goods. Fourthly, If the load capacity is excessive, a change of scale for the RC will be required which can bring complication if a smaller scale was opted for. Another drawback can also be the incompatibility of a larger scale RC car with regards to workspace area.

Applications

- In a workplace or office, comprising of various departments working together and the need to transport physical documents from one station to the other.
- Transporting something as a letter or small parcel from one your letter box outside to you in the house.
- In an environment such as a hospital where biohazard samples requiring minimal human contact need to be transported.
- Warehouse delivery robot

5. CONCLUSIONS

Our goal was to build an autonomous delivery robot capable of transporting material to our desired destination. We chose Arduino MKR1000 connected to the Raspberry Pi 3 on the delivery robot for handling the task. The Arduino MKR1000 has an ultrasonic sensor (HC-SR04) and a Bluetooth module (HC-05) attached to it, and it will read data off the sensors, process it, and send to the Raspberry Pi. The ultrasonic sensor is placed on the container and is used to detect the presence of a letter in the container. The delivery robot needs to detect if it is at the origin or not. To do that, we used a Bluetooth beacon, simply because they are of short-range and easily available. The HC-05 which is wired to the MKR1000 is configured to automatically connect to Bluetooth beacon placed at the destination if it is in the range. This information is sent to the Raspberry Pi, which can then finally detect whether or not it is at the origin. By training and building a model based on Keras supervised learning technique often referred to as behavioral cloning, we have achieved the outcome that lets our robot to drive on its own based on our driving style [11] and to deliver the object as and when required to our desired location.

REFERENCES

[1]"Donkey Car", Docs.donkeycar.com, 2016. [Online]. Available: https://docs.donkeycar.com/.
[2] A. Courses et al., "OpenCV", Opencv.org, 2020. [Online]. Available: https://opencv.org/.
[3]"TensorFlow", TensorFlow, 2020. [Online]. Available: https://www.tensorflow.org/.
[4] A. Satan, "Bluetooth-based indoor navigation mobile system," 2018 19th International Carpathian Control Conference (ICCC), Szilvasvarad, 2018, pp. 332-337.
[5] M. Zaafir Barahim, M. Razvi Doomun and N. Joomun, "Low-Cost Bluetooth Mobile Positioning for Location-based Application," 2007 3rd IEEE/IFIP International Conference in Central Asia on Internet, Tashkent, 2007
[6] Kim YJ, inventor; Samsung Electronics Co Ltd, assignee. I2C communication system and method enabling bi-directional communications. United States patent application US 11/028,319. 2005 Jul 28.
[7] P. Liang, Y. L. Chang and S. Hackwood, "Adaptive self-calibration of vision-based robot systems," in IEEE Transactions on Systems, Man, and Cybernetics, vol. 19, no. 4, pp. 811-824, July-Aug. 1989.
[8] Jain, Aditya Kumar. "Working model of Self-driving car using Convolutional Neural Network, Raspberry Pi and Arduino." 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE, 2018.
[9] Géron, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media, 2019.
[10] M. Currey, "Arduino with HC-05 Bluetooth Module in Slave Mode | Martyn Currey", Martyncurrey.com, 2020. [Online]. Available: http://www.martyncurrey.com/arduino-with-hc-05-bluetooth-module-in-slave-mode/. [Accessed: 24- Oct-2020].
[11] Kaspar Sakmann," Behavioral Cloning — make a car drive like yourself", medium Available at: https://medium.com/@ksakmann/behavioral-cloning-make-a-car-drive-like-yourself-dc6021152713 (2016)
[12] YongChai Tan, BentFei Lew, KhimLeng Tan, KaeVin Goh, KienLoong Lee and ZhiChao Khor, "A new Automated Food Delivery System using autonomous track guided centre-wheel drive robot," 2010 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology, Petaling Jaya, 2010, pp. 32-36.
[13] M. Kocsis, J. Buyer, N. Sußmann, R. Zöllner and G. Mogan, "Autonomous Grocery Delivery Service in Urban Areas," 2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Bangkok, 2017, pp. 186-191.
[14] A. Buchegger, K. Lassnig, S. Loigge, C. Mühlbacher and G. Steinbauer, "An Autonomous Vehicle for Parcel Delivery in Urban Areas," 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, 2018, pp. 2961-2967
[15] S. Senthilkumar, R. Nithya, P. Vaishali, R. Valli, G. Vanitha, L. Ramachanndran," AUTONOMOUS NAVIGATION ROBOT", International Research Journal of Engineering and Technology (IRJET), Volume: 04 Issue: 02 | Feb pp: 2395-0072, 2017