

SHOTGUARD

Qiushi Han, Qianhao Han



OVERVIEW

- **App Name:** **Shotguard**
- **App Description:** a mobile APP and sensing device that not only alerts but also prevents concussions.
- **Track:** Youth Team
- **Category:** Mental health & wellbeing



Concussions

Concussions , a serious concern in hockey, especially for our **young players**. An issue that's been troubling parents, coaches, and players alike. Each year, between **1.7 and 3 million** concussions occur due to sports.

Concussions can be difficult to identify and diagnose, especially in the heat of a game or practice. Young players may not recognize the symptoms or fail to communicate with coaches and parents leading to serious long-term effects on a player's **mental health and well-being**.



SOLUTION – APP & SENSOR

Shotguard

A mobile **APP** and **sensing device** that not only alerts but also **prevents** concussions.

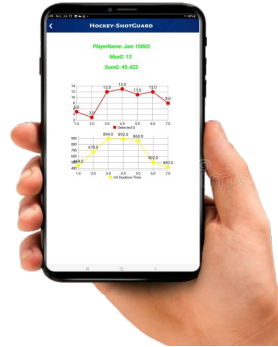
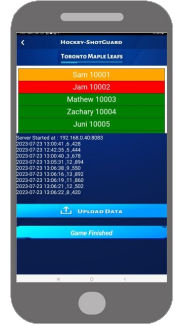
- an inexpensive, compact **IoT** accelerometer sensor attached securely to a player's hockey helmet
- A mobile app developed in **MIT App Inventor**



SOLUTION - CONTINUOUS TRACKING

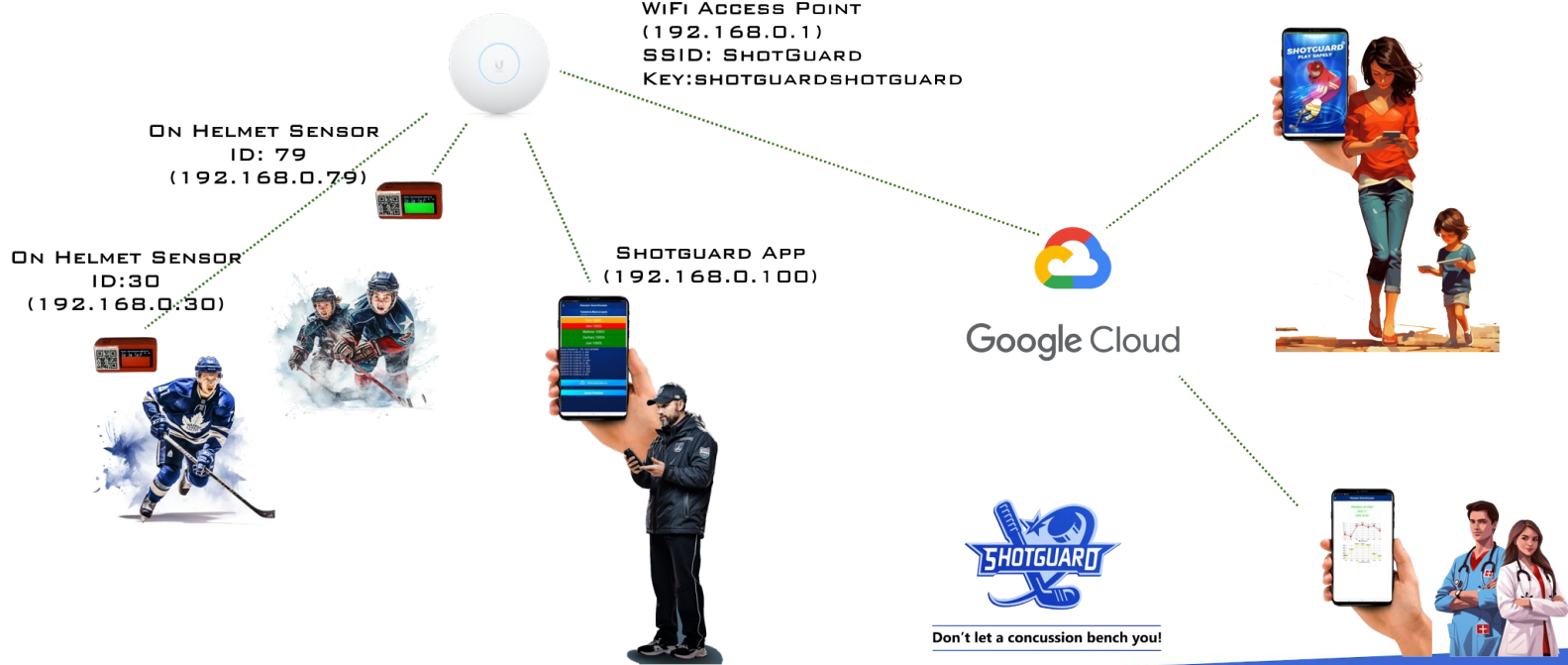
Monitoring

- Constantly monitor the player and track both current and historical motion data
- Alert the user either when a single shock or accumulated shocks reach pre-set thresholds
- The app analyzes this information, identifying patterns that could indicate an increased concussion risk
- Issue immediate alerts about potential hazards and anticipated risks based on the player's data history



SOLUTION - ARCHITECTURE

Networked APP, Sensors & Data Center



MIT APP INVENTOR

```
when UrsaA12TcpServer1 -> MessageReceived
  ClientID Message
do
  call UrsaLimitedStringList1 -> Add
  Text get Message
  call UrsaA12TcpServer1 -> WriteAs
  ClientID get ClientID
  Message get Message
  join "Echo:"
  get Message
  add items to list list
  item get Message
  set global MessageList to
  split text
  get Message
  at
  0
  initialize local clientIDList to
  split text
  get global ClientID
  at
  0
  in
  initialize local id to
  select list item list
  get clientIDList
  index
  0
  in
  initialize local index to
  index in list thing
  list
  get global sensorList
  in
  if
  get index >= 0
  then
  initialize local fileName to
  select list item list
  get global FileList
  index
  get index
  in
  call File1 -> AppendToFile
  text join
  get Message
  "\n"
  fileName get fileName
  initialize local dataList to
  split text
  get Message
  at
  0
  initialize local detectG to
  select list item list
  get dataList
  index
  0
  in
  initialize local durationTime to
  select list item list
  get dataList
  index
  0
  in
  if
  get detectG >= 0
  call TinyDBMaxG -> GetValue
  tag get ID
  valueIfTagNotThere
  0
  then
  call TinyDBMaxG -> StoreValue
  tag get ID
  valueToStore
  get detectG
  initialize local sumG to
  get detectG *
  get DurationTime /
  1000
  in
  set sumG to
  get sumG +
  call TinyDBSumG -> GetValue
  tag get ID
  valueIfTagNotThere
  0
```

```
valueToStore
get detectG
initialize local sumG to
get detectG *
get DurationTime /
1000
in
set sumG to
get sumG +
call TinyDBSumG -> GetValue
tag get ID
valueIfTagNotThere
0
call TinyDBSumG -> StoreValue
tag get ID
valueToStore
get sumG
if
get detectG >= 0
get global StopG
or
get sumG >= 0
get global StopSumG
then
call TableViewPlayer -> SetStyle
selector
join
trnrb-child
get index
0
attribute
background-color
red
value
red
call TableViewPlayer -> ShowTable
else if
get detectG >= 0
get global WarningG
or
get sumG >= 0
get global WarningSumG
then
call TableViewPlayer -> SetStyle
selector
join
trnrb-child
get index
0
attribute
background-color
orange
value
orange
call TableViewPlayer -> ShowTable
```

Dada Collection



MIT APP INVENTOR

Retrieve Player's Information

```
when TableViewerPlayer Click
do
  call UrsAI2TcpServer1 .Stop
  initialize local passData to create empty list
  in initialize local index to get row
  in initialize local id to select list item list get global sensorList
  index get index
  in initialize local playerName to select list item list get global playersList
  index get index
  in initialize local maxG to call TinyDBMaxG .GetValue
  tag get id
  valueIfTagNotThere " "
  in initialize local sumG to call TinyDBSumG .GetValue
  tag get id
  valueIfTagNotThere " "
  in initialize local filename to call TinyDBFileList .GetValue
  tag get id
  valueIfTagNotThere " "
  in
    add items to list list get passData
    item get playerName
    add items to list list get passData
    item get maxG
    add items to list list get passData
    item get sumG
    add items to list list get passData
    item get filename
  open another screen with start value screenName PlayerData
  startValue get passData
```



MIT APP INVENTOR

```
when MatchStart -> Initialize
do
  call MatchStart -> AskForPermission
  permissionName
  Permission (ReadExternalStorage ->
  call MatchStart -> AskForPermission
  permissionName
  Permission (WriteExternalStorage ->
  set global allRecvMessage -> to -> create empty list
  set global playersList -> to -> create empty list
  set global sensorList -> to -> create empty list
  set global sensorList -> to -> call TinyDBSensorPlayerList -> GetTags
  set global playersList -> to -> create empty list
  set global tableViewDataString -> to ->
  for each index from 1
  to length of list list -> get global sensorList ->
  by 1
  do
    initialize local playerName -> call TinyDBSensorPlayerList -> GetValue
    tag -> select list item list -> get global sensorList ->
    index -> get index ->
    valueIfTagNotThere ->
    in -> add items to list list -> get global playersList ->
    item -> get playerName ->
    set global tableViewDataString -> to -> join -> get global tableViewDataString ->
    get playerName ->
    '\n' ->
  set global tableViewPlayerList -> to -> list from csv table text -> get global tableViewDataString ->
  call TableViewPlayer -> Jnit
  layout HA1 ->
  set TableViewPlayer -> FontSize -> to -> 20
  call TableViewPlayer -> SetData
  data -> get global tableViewPlayerList ->
  call TableViewPlayer -> SetStyle
  selector -> table ->
  attribute -> background-color ->
  value -> green ->
  call TableViewPlayer -> SetStyle
  selector -> table ->
  attribute -> color ->
  value -> white ->
  call TableViewPlayer -> ShowTable
  # -> get start value -> playerData ->
  then
  set global fileList -> to -> create empty list
  for each index from 1
  to length of list list -> get global sensorList ->
  by 1
  do
    add items to list list -> get global fileList ->
    item -> call TinyDBFileList -> GetValue
    tag -> select list item list -> get global sensorList ->
```

Displaying Players & Info

```
to length of list list -> get global sensorList ->
by 1
do
  add items to list list -> get global fileList ->
  item -> call TinyDBFileList -> GetValue
  tag -> select list item list -> get global sensorList ->
  index -> get index ->
  valueIfTagNotThere ->
else
  set global fileList -> to -> create empty list
  call TinyDBFileList -> ClearAll
  call TinyDBMaxG -> ClearAll
  call TinyDBSumG -> ClearAll
  # -> length of list list -> get global sensorList -> >= 0 ->
  then
  in for each index from 1
  to length of list list -> get global sensorList ->
  by 1
  do
    initialize local id -> select list item list -> get global sensorList ->
    index -> get index ->
    in initialize local playename -> call TinyDBSensorPlayerList -> GetValue
    tag -> get id ->
    valueIfTagNotThere ->
    in -> initialize local fileName -> to -> join -> get playename ->
    call Clock1 -> FormatDate
    instant -> call Clock1 -> Now
    pattern -> 'yyyyMMdd' ->
    ->
    call Clock1 -> FormatDateTime
    instant -> call Clock1 -> Now
    pattern -> 'hh:mm:ss' ->
    ->
    in -> add items to list list -> get global fileList ->
    item -> get fileName ->
    call TinyDBFileList -> StoreValue
    tag -> get id ->
    valueToStore -> get fileName ->
    call TinyDBMaxG -> StoreValue
    tag -> get id ->
    valueToStore -> 0 ->
    call TinyDBSumG -> StoreValue
    tag -> get id ->
    valueToStore -> 0 ->
  call UriAlertServer1 -> Start
```



MIT APP INVENTOR

Charting Historical Motion Data

```
when File1 .GotText
  text
do
  initialize local readText to get text
in
  initialize local lines to split text
  at "\n"
in
  for each line in list get lines
  do
    initialize local data to split line
    at " "
    in
      add items to list list
      item select list item list
      index 2
      add items to list list
      item get global MaxGList
      select list item list
      index 3
    end
  end
  for each number from 1
  to length of list list
  by 1
  do
    add items to list list
    item get global gDisplayList
    make a list
    get number
    select list item list
    index get number
    get global MaxGList
    add items to list list
    item get global durationDisplayList
    make a list
    get number
    select list item list
    index get number
    get global DurationList
  end
end
call ChartData2D1 .Clear
call ChartData2D1 .ImportFromList
list get global gDisplayList
set ChartData2D1 .Label to "Detected G"
call ChartData2D2 .Clear
call ChartData2D2 .ImportFromList
list get global durationDisplayList
set ChartData2D2 .Label to "Hit Duration Time"
```



MIT APP INVENTOR

```
? when ListView1 .AfterPicking
do
  set global shareFileName to ListView1 . Selection
  call Sharing1 .ShareFile
  file get global shareFileName
  set ListView1 . Visible to false
```

Uploading Data to Google Cloud Data Center



INSTRUCTIONS

Demo Hardware Setup and Preparation

Step 1: Establish a local WiFi network for testing with the fixed IP address "192.168.0.1". Use "ShotGuard" for the SSID and "shotguardshotguard" for the key.

Step 2: Download the Shotguard App onto your Android phone. Ensure the phone's fixed IP is set to 192.168.0.100, and connect your phone to the test WiFi network established in Step 1.

Step 3: Using the Arduino IDE, open the sketch named M5StickCPlus_IMU_ShotGuard_WiFi.ino. Change the IP address in the sketch to 192.168.0.xx (for this example, we use 192.168.0.33). Then, download this sketch onto the Shotguard IoT sensor (specifically, the M5StickC PLUS ESP32-PICO Mini IoT Development Board). This board can be acquired from www.robotshop.ca.

Step 4: Generate and print a QR code that corresponds to the number "xx" from Step 3. For instance, if the IP address is 192.168.0.33, the QR code should read "33".



33



INSTRUCTIONS

Demo Hardware Setup and Preparation

Step 5: Generate and print a QR code for each player. The code should be formatted as "name xxxx". For example, "Sam 10001".

Step 6: Repeat Steps 3 through 5 for every additional sensor you need. Make sure to prepare corresponding QR codes for both the sensors and the players.

By carefully executing these steps, you'll ensure that each player and sensor has a unique QR code for identification during the demonstration.



Sam 10001



INSTRUCTIONS

Using the Shotguard APP



Step 1: Launch the Shotguard App and press the "Attach Sensor" button to begin the registration process for the Shotguard sensor.



Step 2: Scan the QR code on the sensor (e.g., 33 in our example).



Step 3: Press the "Attach Player" button and scan the QR code of the player you're registering (e.g., "Sam 10001").



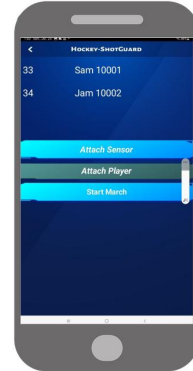
INSTRUCTIONS

Using the Shotguard APP



Step 4: Secure the sensor onto the helmet using the provided pouch.

Step 5: Repeat Steps 2-4 until you've registered all sensors and players on the team.

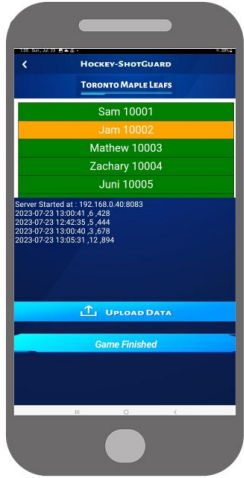


Step 6: Press the "Start Match" button when you're ready to begin the game.

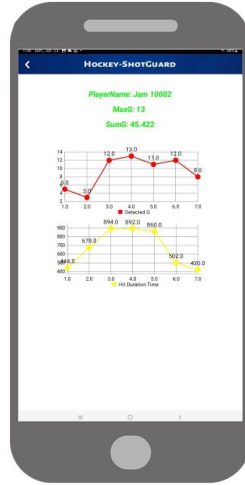
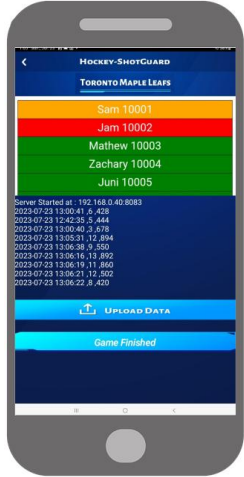


INSTRUCTIONS

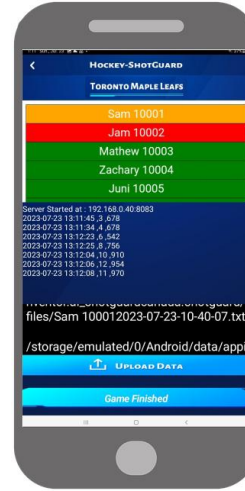
Using the Shotguard APP



Step 7: As the game progresses, alerts will be displayed on the app, color-coded in yellow or red for varying levels of severity.



Step 8: By clicking on a player's name, you can access a detailed view of their historic motion and shock data.



Step 9: Once the match concludes, press "Upload Data" to transfer the collected data to the Google Cloud Data Center for safe storage and future reference. This data can be accessed at any time by authorized users like parents or medical professionals.



FUTURE DEVELOPMENTS

in short future...

- Transition from WiFi to Bluetooth for enhanced device connectivity, thereby eliminating the need for a WiFi access point.
- Implement a robust data storage system to maintain and provide access to historical data, not just from each game, but also long-term player trends.
- Develop a more precise, medically approved concussion alert model, potentially integrating advanced AI modeling, to ensure utmost accuracy in predicting and alerting potential concussions.



LIMITATIONS

Currently, Shotguard is in the early stages of prototyping

- For sensor and app communication, we are using WiFi, but we plan to transition to a Bluetooth connection for future developments. Our IoT sensors are currently set up with fixed IP addresses (e.g., 192.168.0.xxx), where 'xxx' corresponds to the device ID.
- The Android device used must be connected to the same WiFi network and has a fixed IP address of "192.168.0.100". Please note, the device needs to be linked to a Google account as it will upload data files to Google Drive for further processing.
- The thresholds for sensor alerts and warnings are set for demonstration purposes at this time. For real-world applications, these values will need to be refined through extensive studies and clinical trials conducted with professionals in the field.



CONCLUSION

“Shotguard could become a game-changer in preventing concussions.”

-Adam Henrich, former NHL hockey player and head coach of North York Rangers Jr Hockey team



Don't let a concussion bench you!



ACKNOWLEDGEMENTS

We would like to express our deepest appreciation to the following individuals for their invaluable contributions:

- **Adam Henrich**, former NHL hockey player and current head coach of the North York Rangers Jr Hockey team. He generously shared his first-hand knowledge on the impact of concussions on young hockey players' mental health and development, as well as the role of shocks in leading to these concussions.
- **Dr. Serdar Kalaycioglu**, a Research Fellow at the Toronto Metropolitan University. His professional guidance on both hardware and software development played a pivotal role in our project.
- **Bo Wei**, a Senior Engineer at Dr Robot Inc. His expert advice on developing the IoT sensor sketch/program greatly enhanced our technical capabilities.





APPENDIX

SUMMER APPATHON

SHOTGUARD IOT SENSOR

Shotguard IoT Sensor is purchased from www.robotshop.ca

“M5StickC PLUS ESP32-PICO Mini IoT Development Board”

Arduino Sketch/Program: M5StickCPlus_IMU_ShotGuard_WiFi.ino

